The Dissertation Committee for Michael Andrew Scott
certifies that this is the approved version of the following dissertation:

T-splines as a Design-Through-Analysis Technology

Committee:

Thomas J. R. Hughes, Supervisor

Thomas W. Sederberg, Supervisor

Robert L. Taylor

Omar Ghattas

Chad M. Landis

Lexing Ying

T-splines as a Design-Through-Analysis Technology

by

Michael Andrew Scott, B.S.;M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

Dedicated to my wife.

T-splines as a Design-Through-Analysis Technology

Michael Andrew Scott, Ph.D.
The University of Texas at Austin, 2011

Supervisors: Thomas J. R. Hughes
Thomas W. Sederberg

To simulate increasingly complex physical phenomena and systems, tightly integrated design-through-analysis (DTA) tools are essential. In this dissertation, the complementary strengths of isogeometric analysis and T-splines are coupled and enhanced to create a seamless DTA framework. In all cases, the technology developed meets the demands of both design *and* analysis. In isogeometric analysis, the smooth geometric basis is used as the basis for analysis. It has been demonstrated that smoothness offers important computational advantages over standard finite elements. T-splines are a superior alternative to NURBS, the current geometry standard in computer-aided design systems. T-splines can be locally refined and can represent complicated designs as a single watertight geometry. These properties make T-splines an ideal discretization technology for isogeometric analysis and, on a higher level, a foundation upon which unified DTA technologies can be built.

We characterize *analysis-suitable* T-splines and develop corresponding finite element technology, including the appropriate treatment of extraordinary points (i.e., unstructured meshing). Analysis-suitable T-splines form a practically useful subset of T-splines. They maintain the design flexibility of T-splines, including an

efficient and highly localized refinement capability, while preserving the important analysis-suitable mathematical properties of the NURBS basis.

We identify Bézier extraction as a unifying paradigm underlying all isogeometric element technology. Bézier extraction provides a finite element representation of NURBS or T-splines, and facilitates the incorporation of T-splines into existing finite element programs. Only the shape function subroutine needs to be modified. Additionally, Bézier extraction is automatic and can be applied to any T-spline regardless of topological complexity or polynomial degree. In particular, it represents an elegant treatment of T-junctions, referred to as "hanging nodes" in finite element analysis

We then detail a highly localized analysis-suitable h-refinement algorithm. This algorithm introduces a minimal number of superfluous control points and preserves the properties of an analysis-suitable space. Importantly, our local refinement algorithm does not introduce a complex hierarchy of meshes. In other words, all local refinement is done on one control mesh on a single hierarchical "level" and all control points have similar influence on the shape of the surface. This feature is critical for its adoption and usefulness as a design tool.

Finally, we explore the behavior of T-splines in finite element analysis. It is demonstrated that T-splines possess similar convergence properties to NURBS with far fewer degrees of freedom. We develop an adaptive isogeometric analysis framework which couples analysis-suitable T-splines, local refinement, and Bézier extraction and apply it to the modeling of damage and fracture processes. These examples demonstrate the feasibility of applying T-spline element technology to very large problems in two and three dimensions and parallel implementations.

Table of Contents

Abstract v

List of Tables xi

List of Figures xii

Chapter 1. Introduction 1

1.1 Background, motivation, and challenges 1
1.2 T-spline-based isogeometric analysis 6
1.3 Organization of this thesis 10
1.4 Notational conventions 11

Chapter 2. B-spline and NURBS preliminaries 13

2.1 Knot vectors and the B-spline basis 13
2.2 B-spline curves 15
2.3 NURBS curves 16
2.4 NURBS surfaces and solids 18
2.5 The index, parameter, and physical spaces 19
2.6 Knot insertion 23
2.7 Limitations of a NURBS-based framework 24

Chapter 3. T-spline fundamentals 26

3.1 The T-mesh 26
3.2 The T-spline basis 28
3.2.1 Local knot interval vectors 28
3.2.2 The local basis function domain 31
3.2.3 T-spline basis functions 32
3.3 T-spline volumes 33

3.4 The T-spline element structure . . . 37
3.4.1 The local basis function mesh . . . 37
3.4.2 The elemental T-mesh . . . 37
3.4.3 The IEN array . . . 40
3.4.4 Restricting the global basis to the parent element domain . . . 40
3.4.5 The T-spline element geometric map . . . 43

Chapter 4. Bézier extraction **46**
4.1 The element extraction operator and the Bézier element . . . 47
4.2 The Bernstein basis . . . 48
4.3 Computing the element extraction operator for T-splines . . . 51
4.3.1 Univariate extraction . . . 51
4.3.2 Multivariate extraction . . . 53
4.4 A Bézier extraction algorithm for T-splines . . . 57
4.5 Incorporating $\mathbf{C}^e$ into the finite element formulation . . . 60
4.5.1 The element shape function routine . . . 61
4.5.2 Finite element data structures . . . 63
4.5.2.1 Constructing the ID array . . . 64
4.5.2.2 Computing the Bézier mesh . . . 64

Chapter 5. Analysis-suitable T-splines and local refinement **69**
5.1 T-junction extensions . . . 70
5.2 The extension graph . . . 72
5.3 Analysis-suitable definition . . . 72
5.4 The analysis-suitable elemental T-mesh . . . 73
5.5 Partition of unity . . . 74
5.6 Local refinement of analysis-suitable T-splines . . . 74
5.6.1 Local refinement fundamentals . . . 75
5.6.1.1 Basis function refinement . . . 75
5.6.1.2 The refinement operator $\mathbf{M}$ . . . 77
5.6.2 Analysis-suitable nesting theory . . . 77
5.6.3 A local refinement algorithm . . . 80
5.7 Computing the refinement operator $\mathbf{M}$ . . . 89

5.8 Applying analysis-suitable local refinement 92
5.9 Linear independence 103
5.9.1 The T-spline-to-NURBS transform matrix, $\mathbf{N}$ 103
5.9.2 Column reduction 104
5.9.3 The influence graph, G 105

Chapter 6. Extraordinary points **108**
6.1 The T-mesh with extraordinary points 110
6.2 The T-spline basis near extraordinary points 111
6.2.1 Generalized Bézier extraction 111
6.2.2 Smoothing the extraction 114
6.2.2.1 Assembling the constraint equations 116
6.2.2.2 Assembling the fairing system 121
6.3 Analysis-suitability 123
6.4 Patch tests 123

Chapter 7. T-spline-based isogeometric analysis **132**
7.1 Isogeometric fluids analysis 133
7.2 Isogeometric structural analysis 141
7.2.1 Pinched hemisphere 141
7.2.2 Pinched cylinder 142
7.2.3 Hemispherical shell with a stiffener 144
7.3 Isogeometric gradient damage 154
7.3.1 Isotropic damage formulation 154
7.3.1.1 Constitutive modeling 155
7.3.1.2 Implicit gradient damage formulation 157
7.3.2 Numerical results 158
7.4 Isogeometric phase-field fracture 164
7.4.1 Isotropic fracture formulation 166
7.4.2 Adaptive refinement scheme 168
7.4.3 Numerical results 169
7.4.3.1 Dynamic shear loading 169
7.4.3.2 Pressurized cylinder with solid elements 172

Chapter 8. Conclusions and future work **179**

Appendices **182**

Appendix A. T-spline extraction example data **183**
A.1 T-spline control points 183
A.2 Bézier element control points 183
A.3 Bézier element extraction operators 186

Appendix B. Basis function derivatives **191**

Appendix C. T-NURCC subdivision **193**
C.1 The element subdivision matrix 193
C.2 Element definition 196
C.3 Exploiting eigenstructure 199
C.4 The challenge of numerical integration 200
C.5 Subdivision rules 203

Bibliography **207**

List of Tables

1.1 A comparison of C^0 finite elements, NURBS, and T-splines in terms of their potential as a design-through-analysis discretization technology. 6

1.2 Notational conventions used in this dissertation. 12

3.1 The IEN array is constructed using the information from Figure 3.10. The IEN array maps the local basis function number (a) and the element number (e) to the corresponding global control point (A). The local basis function number indexes the T-spline basis functions supported by the element in question. Note that there can be different numbers of T-spline basis functions associated with different elements. 42

4.1 The extraction operators corresponding to the four Bézier elements in Figure 4.5. The highlighted rows correspond to the extraction of N_4 over each element. 53

7.1 A comparison of the "old" local refinement algorithm from [104] and the "new" analysis-suitable local refinement algorithm in terms of Bézier elements and basis functions introduced through refinement. 137

7.2 Bézier meshes used for the L-shaped specimen. 160

7.3 The number basis functions before refinement and the number of elements that were flagged for refinement for each mesh. 172

A.1 The control point ($\mathbf{P}$) coordinates (x, y) and weights (w). 184

A.2 Bézier element control points for elements 1, 10, 11 and 17, as shown in Figure 3.10. The array $a(i, j)$ is defined in (4.9). 185

List of Figures

1.1 Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories. Note that the process of building the model completely dominates the time spent performing analysis. (Courtesy of Michael Hardwick and Robert Clay, Sandia National Laboratories.) 2

1.2 T-splines overcome the shortcomings found in NURBS-based representations of geometry. (a) The hand geometry is modeled with multiple NURBS patches. (b) The location where patches intersect introduces gaps and overlaps. (c) A T-spline model of the hand eliminates gaps and overlaps and is a single, watertight geometry. . . 4

1.3 An entire airplane modeled as a single watertight T-spline. (Courtesy of Sky Greenawalt.) 5

1.4 The design of an analysis-suitable T-spline bumper model. Bumper model courtesy of Juan Santocono. 8

1.5 Modes 7, 9, and 30 for the bumper model. 9

2.1 Quadratic basis functions for open, non-uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$. 15

2.2 B-spline piecewise quadratic curve in $\mathbb{R}^2$. Knot vector and basis functions as in Figure 2.1. 17

2.3 A quadratic NURBS surface generated from $\Xi^1 = \{0, 0, 0, 1, 2, 2, 2\}$ and $\Xi^2 = \{0, 0, 0, 1, 1, 1\}$. a) The physical mesh is comprised of the image of the knot lines under the geometrical mapping. In this case, we have two elements. This is completely analogous to a finite element mesh. b) The control mesh is comprised of the actual control points. Note that only the corner control points lie on the surface. It is extremely important to distinguish between the physical mesh and the control mesh. 20

2.4 The parameter space corresponding to the biquadratic B-spline surface seen in Figure 2.3. The parameter space is the pre-image of the physical mesh. 21

2.5 The index space corresponding to the biquadratic B-spline surface seen in Figure 2.3. 22

2.6 a) There are many instances in which we would like to locally refine an initial NURBS mesh by subdividing an individual element. b) Unfortunately, knot insertion is a global process that necessitates the propagation of the refinement throughout the domain. 25

3.1 The domain $\Omega \subset \mathbb{R}^2$ for a bivariate ($d_p = 2$), cubic ($p = 3$) T-spline. The curved boundaries are exact quarter circles. The hash marks on the left indicate homogeneous Dirichlet boundary conditions. 27

3.2 A T-mesh defining a bicubic T-spline geometry. The large red circles are the T-junctions for this T-mesh. The indexing identifies the T-mesh control points. 28

3.3 A valid knot interval configuration for the bicubic T-mesh in Figure 3.2. The triangles correspond to a knot interval of 0, the squares correspond to a knot interval of $\frac{1}{2}$, and the pentagons correspond to a knot interval of 1. A valid knot interval configuration for a T-mesh element is shown in the element callout. Notice that the knot intervals along opposing sides of the element sum to the same value. 29

3.4 The construction of T-spline basis function N_{18} which is associated with T-mesh vertex $\mathbf{P}_{18}$. Starting at the upper left and going clockwise we have: Inferrence of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function. 34

3.5 The construction of T-spline basis function N_{33} which is associated with the T-junction T-mesh vertex $\mathbf{P}_{33}$. Starting at the upper left and going clockwise we have: Inferrence of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function. 35

3.6 A volumetric T-mesh 36

3.7 Extracting a local knot vector from a volumetric T-mesh 36

3.8 The mapping of T_{33} onto the global T-mesh T. The dashed edges correspond to lines of reduced continuity from T_{33} which are not in T. 38

3.9 The T-spline elements in the elemental T-mesh T_f of T. 39

3.10 T-mesh elements 1, 10, 11, and 17 in the elemental T-mesh and the T-mesh control points whose corresponding T-spline basis functions are non-zero over these T-spline elements. 41

3.11 The mappings between parent, element, and basis function coordinate systems for element 17. For simplicity only some of the mappings are shown. $\tilde{\Phi}^{17}$ maps from the parent coordinate system into the element coordinate system. $\hat{\Phi}^{17}_a$ maps from the element coordinate system of local basis function number a into the basis function coordinate system of global control point $A = \mathrm{IEN}(a, 17)$. The solid dot ($\bullet$) on element 17 shows the rotations incorporated in the mappings. 44

4.1 Bézier extraction for a cubic B-spline curve. The B-spline curve and basis functions are shown on the left. The action of the extraction operator, $\mathbf{C_2}$, for element Ω_2 is illustrated on the right. The transpose of the extraction operator defines the control points, $\mathbf{Q}_2 = \{\mathrm{Q}_{2,I}\}_{I=1}^{4}$, of the Bézier element from the control points, $\mathbf{P}_2 = \{\mathrm{P}_I\}_{I=2}^{5}$ of the B-spline curve. The B-spline basis functions, $\mathbf{N}_2 = \{\mathbf{N}_I\}_{I=2}^{5}$, can be computed over the element by applying the extraction operator to Bernstein polynomial basis functions, $\mathbf{B} = \{\mathrm{B}_I\}_{I=1}^{4}$, defined on the Bézier element. Note that the Bernstein basis is the same for each element. Formation of element arrays can thus be standardized in a shape function routine. 47

4.2 Bernstein basis functions for polynomial degree $p = 1, 2, 3$. 50

4.3 Ordering of the two-dimensional Bernstein polynomials on a bicubic Bézier element. 50

4.4 All Bézier elements in the support of T-spline basis function N_{40} in the elemental T-mesh, T_f (left) and the local basis function mesh, T_{40} (right). 51

4.5 A univariate B-spline basis function, N_4, (solid line) with local knot vector $\Xi = \{0, 1, 2, 3, 4\}$. 52

4.6 The resulting basis functions of the Bézier elements after knot insertion and refinement of the basis functions shown in Figure 4.5. . . 53

4.7 The extraction of a single T-spline basis function. Extraction is decoupled into two univariate extraction operations and then reassembled into a single row of $\mathbf{C}^e$. (The T-spline basis function is the one associated with control point 40, and element e corresponds to element 13 in the elemental T-mesh shown in Figure 3.9.) 55

4.8 The result of extracting $N_{40}(\boldsymbol{\xi}_{40})$ over element e, where $e = 13$. On the top left, the position of element 13 in the local basis function space of $N_{40}(\boldsymbol{\xi}_{40})$ is shown. On the top right, Bézier extraction of N_{40} over element e (see Figure 4.7) generates $\mathbf{c}_{14}^{13\,T}$ where 14 is the local index of global control point 40 for element 13. In other words, $40 = \mathrm{IEN}(14, 13)$. Note that this represents a single row of the element extraction operator $\mathbf{C}^{13}$. On the bottom left, $N_{40}(\boldsymbol{\xi}_{40})$ is plotted. The shaded region is the restriction of $N_{40}(\boldsymbol{\xi}_{40})$ to the domain of element 13. On the bottom right, a close-up of element e is shown which portrays graphically the relationship between the extraction coefficients and Bernstein basis functions. Note that $N_{40}(\boldsymbol{\xi}_{40})|_{13} = \mathbf{c}_{14}^{13\,T}\,\mathbf{B}(\boldsymbol{\xi})$. See equation (4.13). 56

4.9 The ID array maps the degree-of-freedom number (i.e., direction index of the displacement component) and global control point number to the corresponding equation number in the global system. For this example the horizontal and vertical displacements of control points 1, 9, 17, 29, 37, 44, and 51 are specified by boundary conditions. The global equations corresponding to these degrees-of-freedom are removed from the system through the ID array. The LM array is computed as follows: $P = \mathrm{LM}(i, a, e) = \mathrm{ID}(i, \mathrm{IEN}(a, e))$. 66

4.10 The extraction operators and IEN array can be used to construct the Bézier control elements. For each control element e the ◯'s indicate the global T-spline control points which influence the location of Bézier control points, indicated by the •'s. The global control points that influence each control element are determined by the IEN array, and the location of the element control points is computed with the element extraction operator $\mathbf{C}^e$. 67

4.11 Various "meshes" of the T-spline example. 68

5.1 The extended T-mesh, T_{ext}, in the index and physical space. The dotted arrows (in black) represent face extensions and the dashed arrows (in red) represent edge extensions. The T-junctions are denoted by large circles. (a) A T-mesh T with six T-junctions. (b) The extended T-mesh formed from T and the T-junction extensions in index space. The indexing of the knot structure of T is along the bottom and left. (c) The extended T-mesh in physical space. Note that the directionality of each extension is always determined in the index space of the T-mesh. 71

5.2 The extension graph, $\mathrm{E}(\mathsf{T}_{ext})$. (a) The extension graph $E(\mathsf{T}_{ext})$ corresponding to T_{ext} in Figures 5.1b and 5.1c. The five edges in the graph correspond to the five intersections between T-junction extensions in T_{ext}. (b) The T-mesh in Figure 5.1a can be made analysis-suitable by adding the bold dashed edges. The extension graph for this new T-mesh is empty (no edges.) 73

5.3 A T-mesh, T^1, (solid circles and lines) and T-spline space, $\mathcal{T}^1$, is locally refined through the addition of control points and edges (hollow circles and dashed edges). In this case, $\mathsf{T}^1 \subseteq \mathsf{T}^2 \rightarrow \mathcal{T}^1 \subseteq \mathcal{T}^2$. . . 76

5.4 Determining nestedness. (a) An analysis-suitable extended T-mesh T^1_{ext}. (b) An extended T-mesh T^2_{ext} (not analysis-suitable). (c) Superimposing the face extensions from (a) on T^2_{ext} to form $\mathsf{T}^{1\rightarrow 2}_{ext}$. Notice that the only face extension from T^1_{ext} which is visible is in the light gray box. (d) The coupled extension graph for (c). The graph contains edges which implies that $\mathcal{T}^1_s \nsubseteq \mathcal{T}^2$. 79

5.5 Several analysis-suitable local refinements for the example in Figure 5.4. The dashed edges and open circles are T-mesh edges and vertices, respectively, added during local refinement. (a) The T-mesh T^2 from Figure 5.4b. (b) 18 vertices and 19 edges have been added. (c) 7 vertices and 8 edges have been added. (d) 6 vertices and 8 edges have been added. 82

5.6 T-mesh edges which may be inserted during Step 3 of the local refinement algorithm. (a) The T-mesh T^2. (b) The thick black dashed lines are edges which can be inserted into T^2. Notice that each of these edges has a vertex which is a T-junction in T^2 and the corresponding node in the coupled extension graph has non-zero weight (see Figure 5.4d.) The thick black dashed lines are called insertion edges. The thin red dashed lines are examples of T-mesh edges which cannot be inserted to form a refined T-mesh. 83

5.7 Initialization of a simple analysis-suitable local refinement example. (a) The initial analysis-suitable T-mesh before refinement. (b) Four T-mesh elements are subdivided to form T^2. (c) The coupled extended T-mesh $\mathsf{T}^{1\rightarrow 2}_{ext}$ corresponding to (a) and (b). (d) The coupled extension graph $E(\mathsf{T}^{1\rightarrow 2}_{ext})$. The graph is not empty so nesting does not necessarily hold. 84

5.8 The first local refinement iteration for the example in Figure 5.7. (a) The input coupled extension graph. This graph has a weight of 4. (b) The corresponding insertion edges. ΔW and W for each insertion edge are shown in the Table. ΔW measures the change in graph weight after the edge is inserted into T^2. W is the total graph weight after the edge is inserted into T^2. The shaded cells indicate ΔW values which were computed during this iteration. Insertion edge K minimizes the graph weight and is inserted into the T-mesh as a T-mesh edge. Notice that in this case insertion edge E could also have been selected. Both have a ΔW of 2. 86

5.9 The second and final local refinement iteration for the example in Figure 5.7. (a) The input coupled extension graph. This graph has a weight of 2. (b) The corresponding insertion edges. ΔW and W for each insertion edge are shown in the Table. ΔW measures the change in graph weight after the edge is inserted into T^2. W is the total graph weight after the edge is inserted into T^2. Since no ΔW cells are shaded all values are saved from the previous refinement step (see Figure 5.8.) Insertion edge E drives the graph weight to zero and is inserted into the T-mesh as a T-mesh edge to complete the refinement process. 87

5.10 The final refined T-mesh for the example in Figure 5.7. The new edges and vertices are the dashed lines and open circles, respectively. 88

5.11 Computing the refinement coefficient, $m_{A,B}$, corresponding to the exact representation of $N_A^1 \in \mathcal{T}_s^1$ in terms of $N_B^2 \in \mathcal{T}_s^2$. (a) The T-mesh, T_s^1, consists of the solid circles and lines. The topology (control points and edges) added during the topology phase of analysis-suitable local refinement (Steps 1 through 4 in Section 5.6.3) are denoted by the open circles and dashed lines, respectively. The topology phase ensures that $\mathcal{T}_s^1 \subseteq \mathcal{T}_s^2$. (b) The index space of T_s^2. The index domain $\Omega_{A,2}^I = [1,6] \otimes [0,5]$ is the union of the dark and lightly shaded rectangles and the index domain $\Omega_{B,2}^I = [2,6] \otimes [0,4]$ is the lightly shaded rectangle. The origin of the common knot coordinate system is $(1,0)$ in the index space. The knot intervals for the common knot coordinate system are shown in (a) and (b). 90

5.12 An analysis-suitable local refinement framework. 94

5.13 A T-spline container ship hull. The surface is C^2-continuous everywhere. 95

5.14 The regions of the container ship hull where analysis-suitable local refinement will be performed. First, refinement will be performed in the rectangular region followed by highly localized refinement along the curve. 96

5.15 The first iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.) 97

5.16 The second iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.) 98

5.17 The third iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.) 99

5.18 The fourth iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. Notice that analysis-suitable local refinement remains localized to the Bézier elements selected for refinement. (Note: Some of the T-mesh edges are hidden behind the ship hull.) 100

5.19 The fifth iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. Notice that analysis-suitable local refinement remains localized to the Bézier elements selected for refinement. (Note: Some of the T-mesh edges are hidden behind the ship hull.) 101

5.20 The sixth and final iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown below that. The refined final T-mesh is then shown. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.) The final Bézier element mesh is shown on the bottom. The refinements form a nested sequence of C^2-continuous spline spaces. The geometry of the hull is unchanged during the refinements. 102

5.21 A T-mesh and its influence graph 106

6.1 A T-mesh of arbitrary topology. Extraordinary points are denoted by red hollow circles and T-junctions are denoted by red hollow squares. The one-ring neighborhoods are composed of the darkly shaded elements and the two-ring neighborhoods are composed of the dark and lightly shaded elements. The spoke edges are denoted by the thick black lines. 109

6.2 Generalized bicubic Bézier extraction. (a) Face, edge, and vertex points defining an extraordinary element. The Bézier control points are denoted by open circles while the original T-spline control points are denoted by solid circles. Face Bézier points are denoted by a superscript f, edge Bézier points are denoted by a superscript e, and vertex Bézier points are denoted by a superscript v. Notice that the T-spline control points may or may not be extraordinary points. (b) Face points for a Bézier element. Each face point is written in terms of T-spline control points $\mathbf{P}_A$ through $\mathbf{P}_D$ and knot intervals a through f. (c) Edge points corresponding to an edge of an extraordinary element. Each edge point is written in terms of Bézier face points of neighboring Bézier elements and knot intervals a and b. (d) A vertex point corresponding to a corner of an extraordinary element. Each vertex point is written in terms of all adjacent face points and spoke edge knot intervals. 112

6.3 Generalized Bézier extraction. (top) The T-spline basis functions corresponding to the two extraordinary points in Figure 6.1. These basis functions *do not* lie in a portion of the T-mesh with a rectangular grid topology. The Bézier mesh defining each basis function is also plotted. (bottom) The T-spline basis function, corresponding to the valence five extraordinary point in Figure 6.1. The lack of smoothness of the basis function is evident. 115

6.4 The action of the local-to-global index map, $I(i, \alpha, \beta)$. 117

6.5 The i^{th} quadrant of a two-ring neighborhood around an extraordinary point (top) and a close-up of the extraction coefficients involved in the G^1 constraint equations (bottom). The $\mathbf{w}^i$, $\mathbf{x}^i$, $\mathbf{y}^i$, and $\mathbf{z}^i$ represent Bézier elements, the a^i represent knot intervals, and the $c^i_{\alpha\beta}$ denote extraction coefficients used in the G^1 constraint equations (6.18) – (6.23). 118

6.6 The smoothed G^1-continuous T-spline basis function corresponding to the valence five extraordinary point in Figure 6.1. Compare this result to the basis function on the bottom of Figure 6.3. 122

6.7 Standard patch tests. All patch tests are performed on a unit square with a Poisson's ratio of 0.3 and a Young's modulus, E, of one. The box on the right specifies the boundary conditions for rigid translation, rigid rotation, stretch, and shear patch tests. 125

6.8 The T-mesh used for the patch tests in Figure 6.7. The knot intervals associated with the edges intersected by the dotted lines are varied resulting in the Bézier meshes in Figure 6.9. 126
6.9 The Bézier meshes, corresponding to the T-mesh in Figure 6.8, for $k = 1, 5, 10$ and 100. 127
6.10 Results for the rigid translation patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Constant displacement profiles in the x and y direction are reproduced exactly and all stress states are zero. 128
6.11 Results for the rigid rotation patch test. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y directions are reproduced exactly and all stress states are zero. 129
6.12 Results for the stretch patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y direction are reproduced exactly and σ_{xx} and σ_{yy} are constant while all other stress states are zero. 130
6.13 Results for the shear patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y direction are reproduced exactly and σ_{xy} is constant while all other stress states are zero. 131

7.1 Reaction diffusion problem. Problem description and data. 134
7.2 Reaction diffusion problem. Sequence of T-spline meshes. 135
7.3 Reaction diffusion problem. Results for $p = 1$. 136
7.4 Reaction diffusion problem. Results for $p = 2$. 137
7.5 Reaction diffusion problem. Results for $p = 3$. 138
7.6 Advection diffusion problem, $\theta = 45^\circ$. Problem description and data. 139
7.7 Advection skew to the mesh, $\theta = 45^\circ$. Results for $p = 3$. The Bézier meshes are on the left and the corresponding solutions are on the right. 140
7.8 Shell Obstacle Course. Pinched hemisphere problem description and data. 142
7.9 Shell Obstacle Course. Sequence of T-spline meshes for pinched hemisphere for $p = 2$. 143
7.10 Shell Obstacle Course. Pinched hemisphere with $p = 5$, NDOF = 1524: (a) Physical mesh and (b) Displacement contours in direction of inward directed point load on deformed configuration (scaling factor of 30 used). 144

7.11 Shell Obstacle Course. Pinched hemisphere displacement convergence. 145
7.12 Shell Obstacle Course. Pinched cylinder problem description and data. 146
7.13 Shell Obstacle Course. Sequence of T-spline meshes for pinched cylinder for $p = 2$. 146
7.14 Shell Obstacle Course. Pinched cylinder with $p = 5$, NDOF = 1260: (a) Physical mesh and (b) Displacement contours on deformed configuration (scaling factor of 3×10^6 used). 147
7.15 Shell Obstacle Course. Pinched cylinder displacement convergence. 147
7.16 Hemispherical shell with stiffener. Problem description from Rank et al. [91]. 148
7.17 Hemispherical shell with stiffener. Initial mesh, NDOF = 360: (a) Coarse mesh and (b) Detail of stiffener. 149
7.18 Hemispherical shell with stiffener. Refined meshes. 149
7.19 Hemispherical shell with stiffener. Detail of refinement for finest mesh (NDOF = 4248). 150
7.20 Finest mesh (NDOF = 4248): (a) Vertical displacement contours (scaling factor of 500 used) and (b) von Mises stress contours, detail of stiffener. 151
7.21 Hemispherical shell with stiffener. Convergence of displacement and von Mises stress at various points to benchmark solution [91]. . 152
7.22 Hemispherical shell with stiffener. Convergence of displacement and von Mises stress at various points to benchmark solution [91]. . 153
7.23 Solid domain Ω with boundary $\partial\Omega$. 155
7.24 L-shaped specimen. The thickness of the specimen is 200 mm. . . . 160
7.25 Bézier meshes for the L-shaped specimen. 161
7.26 Smooth (C^2) T-spline basis function centered around the reentrant corner of the L-shaped domain. 162
7.27 Mesh convergence studies using the cubic T-spline meshes in Figure 7.25 for the second-order (a), fourth-order (b) and sixth-order (c) damage formulations, and for the nonlocal formulation (d). . . . 163
7.28 Force-displacement results for the L-shaped specimen using the nonlocal formulation and d-th order gradient formulations. All results are obtained using Mesh 3. 164
7.29 Isolines for the damage parameter $\omega = 0.8$ in the L-shaped specimen at $u = 2$ mm as computed on Mesh 3 by the second-order formulation and the sixth-order formulation. Displacements are amplified by a factor of 15. 165

7.30 A contour plot of the sixth-order damage field corresponding to Mesh 3. 166
7.31 (a) Schematic representation of a solid body Ω with internal discontinuity boundaries Γ. (b) Approximation of the internal discontinuity boundaries by the phase-field $c(x,t)$. The model parameter ϵ controls the width of the failure zone. 167
7.32 The geometry and boundary conditions for the dynamic shear loading example. The crack is modeled by an actual discontinuity in the mesh with a zero radius crack tip. The load is applied as a velocity condition that is ramped up from 0 to 16.5 m/s in one microsecond and then held constant for the duration of the simulation. 170
7.33 Kalthoff mesh refinement results. Mesh 1 is a 128 x 128 cubic T-spline mesh. Bézier elements were flagged for refinement if $c < 0.8$ at any quadrature point inside the element and $h = \sqrt{a} > 1.94 \times 10^{-4}$ m where a is the element area. 172
7.34 The first four meshes in the local refinement sequence. The elements in red are those that were selected to be refined at each step. . 173
7.35 The (a) elastic strain energy and (b) dissipated energy for the sequence of refined meshes shown in Figure 7.33. The reference mesh is a uniform quadratic NURBS mesh with 1024×1024 Bézier elements. 174
7.36 Geometry and symmetry conditions for the pressure vessel simulation. The mesh is a three-dimensional thickened T-spline. 174
7.37 The final mesh for the pressurized cylinder example problem. The volumetric mesh was constructed by thickening a mid-surface mesh. The refinement was performed using the adaptive refinement scheme describe in Section 7.4.2 which resulted in a final mesh containing 862,100 basis functions. 175
7.38 The results of the pressurized cylinder example. The phase-field is shown. 176
7.39 A post-processed plot of the pressure vessel example at $t = 1.76 \times 10^{-3}$ s. The displacements have been scaled by a factor of 5 and areas of model where $c < 0.05$ have been removed from the plot. Displacement is measured in meters. 177
7.40 Cross section views showing the three-dimension profile of the crack surfaces. 178

C.1 A T-mesh after one T-NURCC subdivision step. The position of the vertices represented by hollow circles is computed as a convex combination of existing control points (solid circles.) The subdivision process remains localized to the T-mesh elements in the two-ring neighborhood of the extraordinary point. 194

C.2 An extraordinary element (shaded element) after one application of T-NURCC subdivision. 195

C.3 An extraordinary element defined through T-NURCC subdivision. . 197

C.4 The L^2 and H^1 norms of the error for the stretch patch test described in Figure 6.7 for increasing gauss rule over each one-ring extraordinary element. All non-extraordinary elements are integrated using four-point gauss quadrature. 201

C.5 The L^2 and H^1 norms of the error for the stretch patch test described in Figure 6.7 for exact integration of each subelement on subdivision level n. Four-point gauss quadrature is used in all cases. 202

C.6 Notational conventions used to describe T-NURCC subdivision. Due to symmetry only one quadrant of a two-ring neighborhood is shown. The vertices labeled $\mathbf{P}^i_{jk}$ denote initial control points before subdivision is performed. The vertices labeled $\mathbf{Q}^i_{jk}$ denote control points resulting from subdivision. The knot intervals are labeled a^i_j. The dashed lines represent edges added during a subdivision step. 204

Chapter 1

Introduction

1.1 Background, motivation, and challenges

The realistic simulation of increasingly complex physical phenomena and systems places a premium on tightly integrated design-through-analysis (DTA) tools. The current state of the art in engineering design and analysis is built on disparate geometric foundations. This leads to many translational difficulties which effect the efficiency and accuracy of the entire process. The anatomy of the DTA process has been studied by Ted Blacker, Manager of Simulation Sciences at Sandia National Laboratories, and is summarized in Figure 1.1 along with the breakdown in the percentage of time devoted to each task. Note that at Sandia mesh generation accounts for only 20 percent of overall analysis time, whereas creation of the analysis-suitable geometry requires about 57 percent, and only 23 percent of overall time is actually devoted to analysis. The approximate 80/20 modeling/analysis ratio is a very common industrial experience and indicates a lack of compatibility in the fundamental technologies underpinning modern DTA methodologies.

Isogeometric analysis has emerged as an important alternative to traditional engineering design and analysis methodologies. Isogeometric analysis was introduced in [57] and later described in detail in [38]. In isogeometric analysis, the smooth geometric basis is used as the basis for analysis and, consequently, the finite element mesh *is* an exact representation of the geometry. By "smooth" we usually mean that interelement continuity may be C^n, $0 \leq n < p$, where p is the

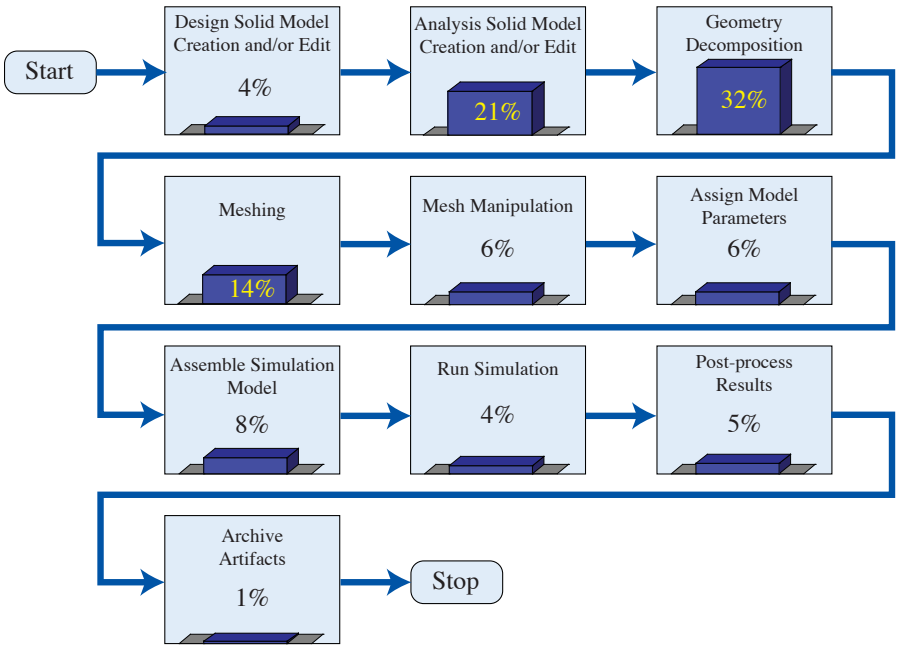


Figure 1.1: Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories. Note that the process of building the model completely dominates the time spent performing analysis. (Courtesy of Michael Hardwick and Robert Clay, Sandia National Laboratories.)

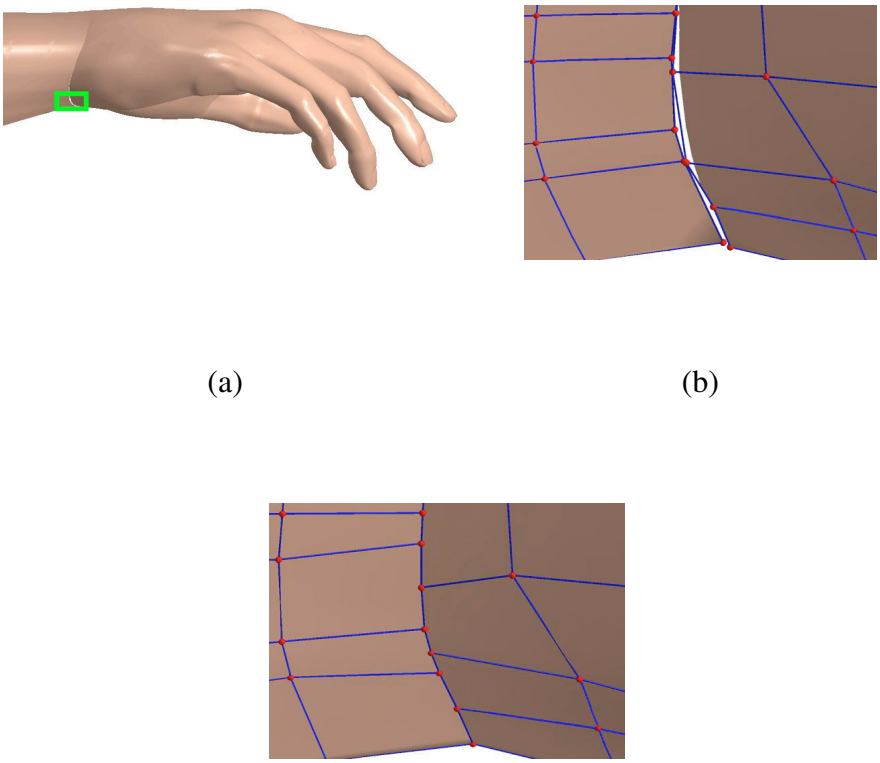
polynomial degree of the basis.

Most of the early developments in isogeometric analysis focused on establishing the behavior of the smooth bivariate and trivariate NURBS basis in analysis since NURBS geometry underpins almost all major design packages. It was demonstrated that smoothness offers important computational advantages over standard finite elements [40,47]. Areas of application of NURBS-based isogeometric analysis include turbulence [2,8,10,15], fluid-structure interaction [11–13,121], incompressibility [4,5,46], structural analysis [39,40], plates and shells [18–20,45,64,65], phase-field analysis [53,54], large deformation with mesh distortion [72], shape op-

timization [84,85,89,118], and electromagnetics [28]. This success has in turn stimulated efforts within the Computer Aided Geometric Design (CAGD) community to develop and integrate analysis-suitable geometric technologies and isogeometric analysis [1, 36, 37, 66, 79, 80, 119, 120].

While the smoothness of NURBS make them useful in isogeometric analysis, NURBS are severely limited by the fact that they are four sided. This hampers their utility as a *design* tool. In traditional NURBS-based design, modeling a complicated engineering design often requires hundreds, if not thousands, of NURBS patches which are usually discontinuous across patch boundaries. Also, almost all NURBS models use trimming curves. Consequently, a global geometric discretization, based on NURBS, is usually not suitable as a basis for analysis. As an example of the problems inherent in NURBS, Figure 1.2a shows a hand model created using multiple NURBS patches [106]. The NURBS construction engenders a gap in the model as shown in Figure 1.2b. These gaps and overlaps must be repaired before an analysis model can be created.

T-splines were introduced in [106]. T-splines can model complicated designs as a single, watertight geometry. With T-splines, the same hand model can be created without any gaps or overlaps as shown in Figure 1.2c. Additionally, NURBS are T-splines so existing technology based on NURBS extends to T-splines. Any trimmed NURBS model can be represented by a watertight trimless T-spline [105] and multiple NURBS patches can be merged into a single watertight T-spline [61, 106]. Unlike NURBS, T-splines can be locally refined [104]. These properties make T-splines an ideal technology for isogeometric discretizations.



(a) (b)

(c)

Figure 1.2: T-splines overcome the shortcomings found in NURBS-based representations of geometry. (a) The hand geometry is modeled with multiple NURBS patches. (b) The location where patches intersect introduces gaps and overlaps. (c) A T-spline model of the hand eliminates gaps and overlaps and is a single, watertight geometry.

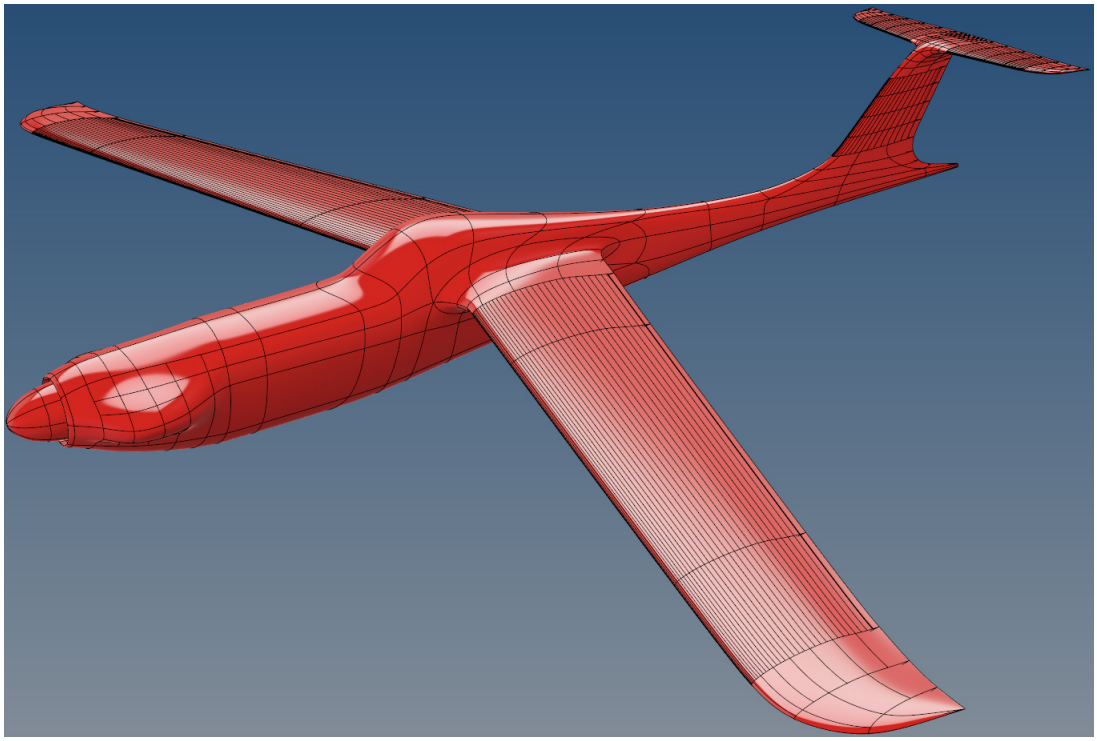


Figure 1.3: An entire airplane modeled as a single watertight T-spline. (Courtesy of Sky Greenawalt.)

Figure 1.3 shows an entire airplane modeled as a *single* watertight T-spline surface. Notice the smooth transitions between the airfoils, which were designed to exact engineering specifications, and the fuselage. To model this same airplane using NURBS, many patches would be required.

Table 1.1 lists several of the most critical and/or desirable properties of a design-through-analysis discretization technology. Traditional C^0 finite elements, NURBS, and T-splines are compared. Notice that several of the properties such as linear independence, satisfaction of patch tests, and simplicity of implentation in FEA codes are specific to analysis while other properties such as a trimmed

to trimless conversion option, current use in design, and forwards and backwards compatibility with NURBS are specific to design. It is interesting to note that the largest group of properties are required or beneficial to both design *and* analysis. These are water tightness, a partition of unity property, affine covariance, local refinement, accommodation of extraordinary points (i.e., unstructured meshing), exact representation of conic sections, and higher-order smoothness. As will be demonstrated in this dissertation, the strengths of T-splines meet the needs of both design and analysis.

	C^0 FE	NURBS	T-splines
Water tight	✓		✓
Linearly independent	✓	✓	✓
Partition of unity property	✓	✓	✓
Affine covariance	✓	✓	✓
Pass standard patch tests	✓	✓	✓
Locally refineable	✓		✓
Accommodate extraordinary points	✓		✓
Trimless option			✓
Simply implemented in FEA codes	✓	✓	✓
Used in design		✓	✓
Forwards and backwards compatible with NURBS		✓	✓
Exact representation of conic sections		✓	✓
Higher-order smoothness		✓	✓

Table 1.1: A comparison of C^0 finite elements, NURBS, and T-splines in terms of their potential as a design-through-analysis discretization technology.

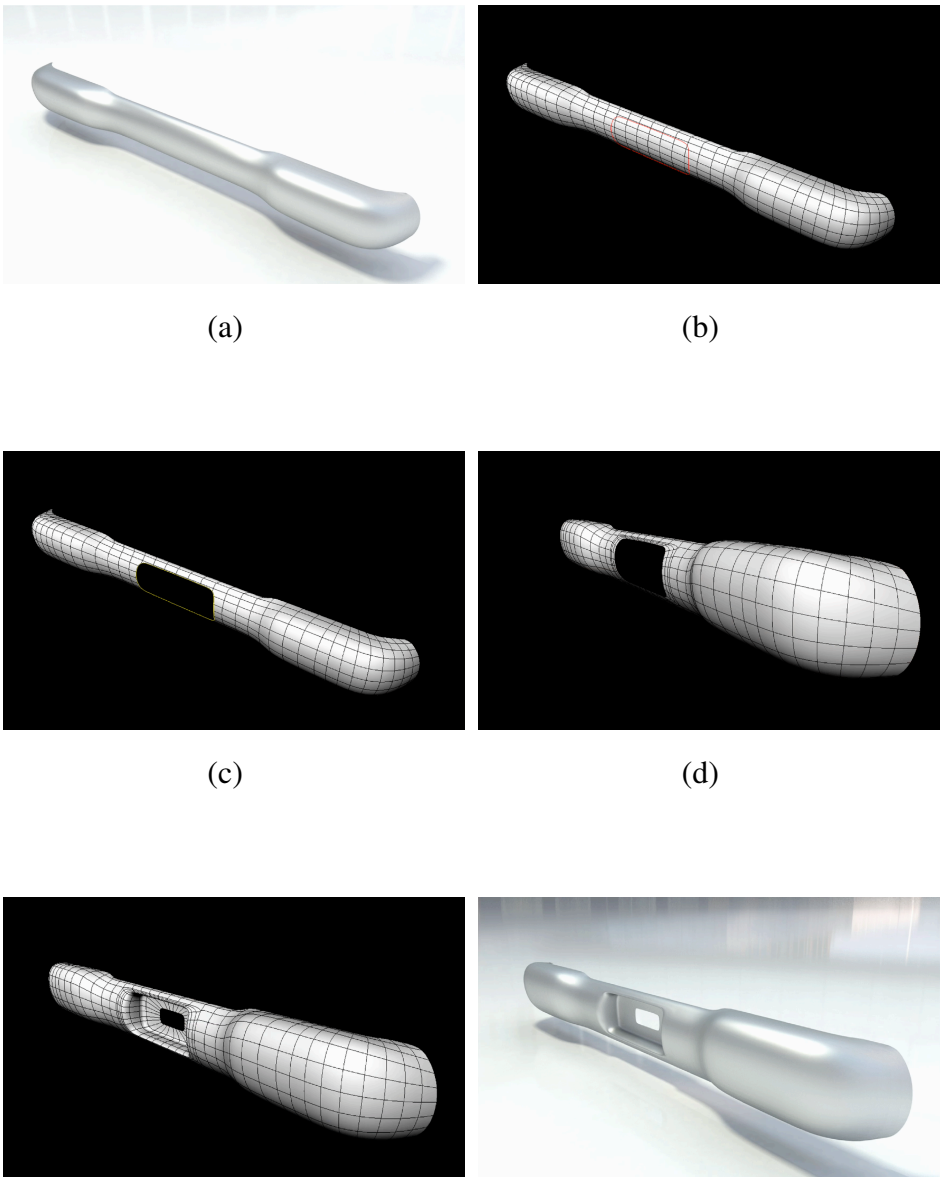
1.2 T-spline-based isogeometric analysis

In this dissertation, fundamental and complementary T-spline and isogeometric analysis capabilities are developed to create a unified DTA technology. We call this *T-spline-based Isogeometric Analysis*. The primary contributions of this

dissertation are:

- Development of Bézier extraction as a unifying paradigm underlying isogeometric finite element technology.
- Characterization of analysis-suitable T-splines in analysis.
- Formulation and implementation of a localized analysis-suitable refinement algorithm.
- Formulation and implementation of analysis-suitable T-splines which accommodate extraordinary points (i.e., unstructured meshes).
- Development and implementation of an efficient adaptive isogeometric framework which couples analysis-suitable T-splines, Bézier extraction, and local refinement, and application of this strategy to problems of implicit gradient damage and phase-field modeling of brittle fracture.

An example which effectively synthesizes the basic ideas and demonstrates, on a high level, the potential of T-spline-based isogeometric analysis is shown in Figures 1.4 and 1.5. The car bumper model was generated using the T-spline plug-in for Rhino [114] and the back-end T-Tools libraries [113]. In Figure 1.4a a bi-cubic NURBS is used to model the general shape of the bumper. To add features, such as holes, it is common to use trimming curves. Figures 1.4b and 1.4c show the placement of a trimming curve and the resulting geometry once the trim is applied. In a DTA environment, the application of trimming curves destroys the analysis-suitable nature of the geometry. The geometric basis no longer describes the geometry and cannot be used directly in analysis.

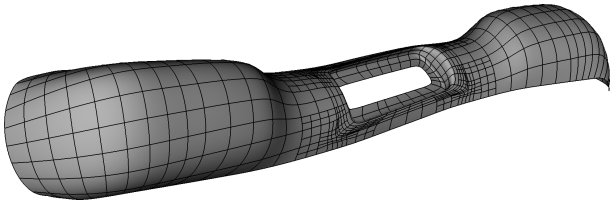


(a) (b)

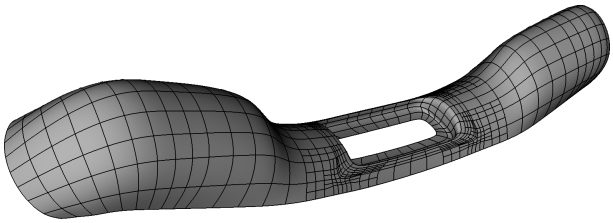
(c) (d)

(e) (f)

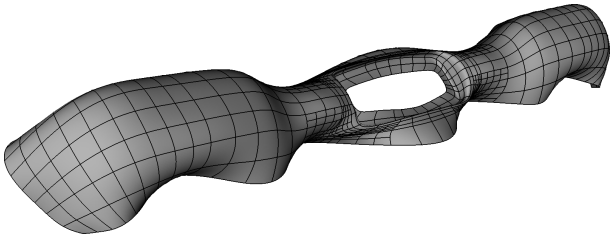
Figure 1.4: The design of an analysis-suitable T-spline bumper model. Bumper model courtesy of Juan Santocono.



Mode 7



Mode 9



Mode 30

Figure 1.5: Modes 7, 9, and 30 for the bumper model.

Using T-splines, however, it is possible to overcome the trimming problem by converting a trimmed T-spline into an untrimmed, watertight, analysis-suitable T-spline. The details of this process are described in [105]. The conversion process first modifies the topology of the T-spline to accomodate any trimming curves. A fitting procedure is then used to match the T-spline surface to the trimming curve. Figure 1.4d shows the untrimmed T-spline which matches the original trimmed T-spline upon completion of the conversion process. This untrimmed T-spline is now analysis-suitable. Additional modeling generates the final bumper geometry shown in Figures 1.4e and 1.4f.

The final model of the bumper consists of 876 cubic T-spline shell elements with 705 control points. No intermediate geometry clean-up or meshing step was employed. The free-free eigenvalues were calculated. The calculations were performed directly on the T-spline model in LS-DYNA. See [18, 19] for further details. The seventh, ninth and thirtieth modes are shown in Figure 1.5.

1.3 Organization of this thesis

Chapter 2 provides a brief overview of B-splines and NURBS curves, surfaces, and solids. Emphasis is placed on those ideas which serve as the foundation for more advanced T-spline concepts introduced in later chapters. Chapter 3 develops fundamental T-spline technology focusing on the T-mesh, the T-spline basis, and the T-spline finite element structure. Chapter 4 introduces Bézier extraction as a finite element technology. Bézier extraction allows common Computer Aided Design (CAD) representations (including T-splines) to serve directly as a basis for conventional Finite Element Analysis (FEA). Chapter 5 introduces analysis-suitable T-splines and develops a new local refinement algorithm. Analysis-suitable T-splines form a practically useful subset of T-splines which maintain the important

mathematical properties of NURBS while providing an efficient and highly localized refinement capability. In Chapter 6, analysis-suitable T-splines are extended to encompass T-splines of arbitrary topological complexity. This is accomplished by introducing a simple isogeometric element for analysis-suitable T-splines near extraordinary points which does not depend upon a recursive subdivision procedure. In Chapter 7, T-splines are applied to various applications in isogeometric analysis. Emphasis is placed on the discretization of damage and fracture processes, and problems are selected which demonstrate the various beneficial aspects of T-spline element technology. Finally, in Chapter 8, we draw conclusions and outline directions for future research.

A favorable side effect of undertaking a thesis topic of this scope is the opportunity and necessity of collaborating with many stellar researchers across multiple disciplines. These collaborations have contributed directly to both the depth and breadth of this dissertation and will continue to enrich and broaden T-spline and isogeometric analysis technology in the future. As a result, this dissertation is a compilation of much, but not all, of the information contained in [9, 18, 23, 24, 69, 71, 99–101, 115–117, 120]. In some cases, only those results which demonstrate the properties and potential of T-splines and isogeometric analysis are included and the interested reader is referred to the corresponding papers for additional details.

1.4 Notational conventions

One of the biggest challenges, when synthesizing ideas from disparate disciplines, is establishing coherent and unified notational conventions. We have gone to great lengths to establish the simplest possible conventions and to faithfully adhere to them. We hope this makes the process of learning T-splines and isogeometric analysis a little less daunting and facilitates the communication of the basic ideas.

The most common notational conventions are listed in Table 1.2.

$A, B, C, \ldots$	Global basis function or control point index
$a, b, c, \ldots$	Local basis function or control point index
i, j, k	Tensor product component index
e	Element index
N (or R, if rational)	Global basis function (NURBS or T-spline)
B	Bernstein basis function
$\mathbf{P}$	Global control point (NURBS or T-spline)
$\mathbf{Q}$	Bézier control point
w	Control weight
Ξ	Knot vector
$\Delta\Xi$	Knot interval vector
T	T-mesh
T_s	Analysis-suitable T-mesh
T_{elem}	Elemental T-mesh
T_{ext}	Extended T-mesh
$\mathcal{T}$	T-spline space
$\mathcal{T}_s$	Analysis-suitable T-spline space
$\mathbf{x}^e$	T-spline element mapping
Ω^e	Physical T-spline element domain
$\tilde{\Omega}$	Parent element domain
$\tilde{\boldsymbol{\xi}}$	Parent coordinate
n	Number of control points or basis functions
p	Polynomial degree
d_s	Number of spatial dimensions
d_p	Number of parametric dimensions
$\mathbf{C}^e$	Element extraction operator
$\mathbf{M}$	Refinement operator

Table 1.2: Notational conventions used in this dissertation.

Chapter 2

B-spline and NURBS preliminaries

As background for our discussion of T-splines, we provide a brief overview of B-spline and NURBS curves, surfaces, and solids focusing on those developments which are critical to isogeometric analysis and T-splines. For a more detailed description of the geometric concepts see [88, 96, 103].

The cases considered here consist of a single NURBS patch. However, the generalization to the multi-patch NURBS case is straightforward. It just involves a transformation between the control point indices of each patch and the corresponding global control points as described in [38].

2.1 Knot vectors and the B-spline basis

A *knot vector* is a non-decreasing sequence of real numbers that indicate parameter values at which continuity is not C^∞. A knot vector is denoted $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ where $\xi_A \in \mathbb{R}$ is the A^{th} knot, p is the polynomial degree of the B-spline basis functions, and n is the number of basis functions. ***B-spline basis functions*** for a given degree, p, are defined recursively by the Cox-de Boor recursion formula as follows:

$$N_{A,0}(\xi) = \begin{cases} 1 & \xi_A \leq \xi < \xi_{A+1} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

$$N_{A,p}(\xi) = \frac{\xi - \xi_A}{\xi_{A+p} - \xi_A} N_{A,p-1}(\xi) + \frac{\xi_{A+p+1} - \xi}{\xi_{A+p+1} - \xi_{A+1}} N_{A+1,p-1}(\xi). \qquad (2.2)$$

The de Boor algorithm [48] provides a standard method for evaluating a B-spline.

B-splines possess the following properties:

- *Partition of unity*:

$$\sum_{A=1}^{n} N_{A,p}(\xi) = 1, \quad \xi \in [\xi_1, \xi_{n+p+1}] \qquad (2.3)$$

- *Pointwise nonnegativity*:

$$N_{A,p}(\xi) \geq 0, \quad A = 1, 2, \ldots, n \qquad (2.4)$$

- *Linear independence*:

$$\sum_{A=1}^{n} c_A N_{A,p}(\xi) = 0 \Leftrightarrow c_B = 0, \quad B = 1, 2, \ldots, n \qquad (2.5)$$

- *Compact support*:

$$\{\xi \,|\, N_{A,p}(\xi) > 0\} \subset [\xi_A, \xi_{A+p+1}] \qquad (2.6)$$

- *Control of continuity*: If ξ_A has multiplicity k (i.e., $\xi_A = \xi_{A+1} = \ldots = \xi_{A+k-1}$), then the basis functions are C^{p-k}-continuous at ξ_A. When $k = p$, the basis is C^0 and interpolatory at that location.

These features are very useful in a finite element context. The first four properties ensure a well conditioned and sparse stiffness matrix. The fifth property allows continuity to be reduced to better resolve steep gradients [39], although in general higher continuity leads to superior accuracy per degree-of-freedom compared with

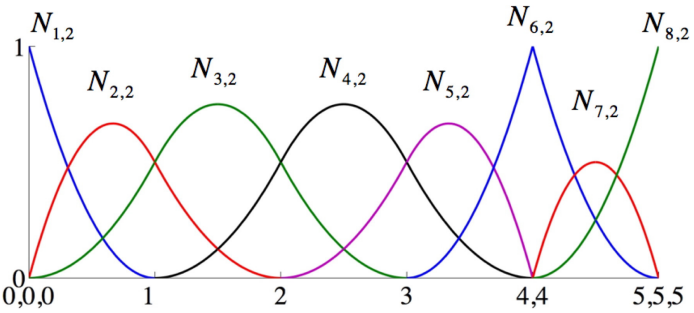


Figure 2.1: Quadratic basis functions for open, non-uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.

C^0-continuous bases [2,40,59]. Additionally, one can use B-splines to build a basis that spans the same space as classical p-version finite elements (that is, a basis of order p that is C^0 across element boundaries). This is the well-known Bernstein basis [78].

An example of a quadratic B-spline basis for $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$ is shown in Figure 2.1. The basis is interpolatory at $\xi = 0, 4, 5$ due to knot repetition. Otherwise, the basis is $C^{p-1} = C^1$ across element boundaries. If the first and last knot in a knot vector are repeated at least p times the knot vector is called an *open* knot vector.

2.2 B-spline curves

A *B-spline curve* of degree p in $\mathbb{R}^{d_s}$ is defined by a set of B-spline basis functions, $\mathbf{N} = \{N_{A,p}\}_{A=1}^{n}$, and control points, $\mathbf{P} = \{\mathbf{P}_A\}_{A=1}^{n}$, as

$$\mathbf{C}(\xi) = \sum_{A=1}^{n} \mathbf{P}_A N_{A,p}(\xi) \tag{2.7}$$

Important properties of B-spline curves are:

- *Affine Covariance*: An affine transformation of a B-spline curve is obtained by applying the transformation to its control points.

- *Convex Hull*: A B-spline curve lies within the union of all convex hulls formed by $p + 1$ contiguous control points (see [96] for the relationship between the convex hull and the polynomial order of the curve).

- *Variation Diminishing*: A B-spline curve in $\mathbb{R}^{d_s}$ cannot cross an affine hyperplane of codimension 1 (e.g., a line in $\mathbb{R}^2$, plane in $\mathbb{R}^3$) more times than does its control polygon [88].

In addition, B-spline curves inherit all of the continuity properties of their underlying bases. This is illustrated in Figure 2.2a, where we built a B-spline curve from the basis shown in Figure 2.1. At the spatial location corresponding to parameter value $\xi = 4$, the B-spline curve is only continuous. The B-spline curve interpolates the control point $\mathbf{P}_6$ at this location. The use of open knot vectors ensures that the first and last control points, $\mathbf{P}_1$ and $\mathbf{P}_8$, are interpolated as well.

2.3 NURBS curves

A NURBS (Non-Uniform Rational B-Spline) is defined by a knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$, a set of rational basis functions $\mathbf{R} = \{R_{A,p}\}_{A=1}^{n}$, and a set of control points $\mathbf{P} = \{\mathbf{P}_A\}_{A=1}^{n}$ as

$$\mathbf{C}(\xi) = \sum_{A=1}^{n} \mathbf{P}_A R_{A,p}(\xi). \tag{2.8}$$

The NURBS basis functions are defined as

$$R_{A,p}(\xi) = \frac{w_A N_{A,p}(\xi)}{W(\xi)} \tag{2.9}$$

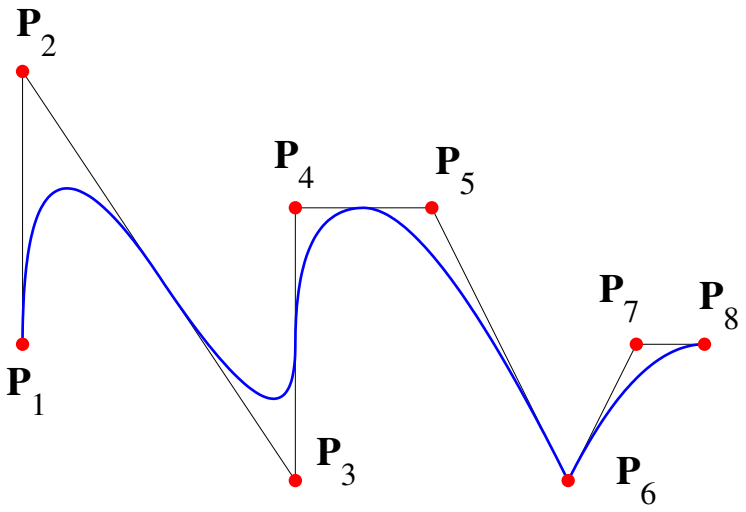


Figure 2.2: B-spline piecewise quadratic curve in $\mathbb{R}^2$. Knot vector and basis functions as in Figure 2.1.

f

where

$$W(\xi) = \sum_{B=1}^{n} w_B N_{B,p}(\xi) \tag{2.10}$$

is the *weight function* and w_B is the weight corresponding the the B^{th} NURBS basis function.

For more efficient computation, a rational curve in $\mathbb{R}^n$ can be represented by a polynomial curve in the *projective space* $\mathbb{P}^n$. As an example, if $\mathbf{P}_A$ is a control point of a NURBS curve then the corresponding homogeneous control point in projective space is $\tilde{\mathbf{P}}_A = \{w_A \mathbf{P}_A, w_A\}^T$. Thus, given a NURBS curve defined in $\mathbb{R}^n$ by (2.8) the corresponding B-spline curve defined in $\mathbb{P}^n$ is

$$\mathbf{C}(\xi) = \sum_{A=1}^{n} \tilde{\mathbf{P}}_A N_{A,p}(\xi).$$

Working in the projective coordinate system allows the algorithms which operate on B-splines to be applied to NURBS.

2.4 NURBS surfaces and solids

To maintain single-index notation, which is standard for T-splines, we introduce a mapping, $\tilde{A}$, between the tensor product space and the global indexing of the basis functions and control points. Let $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $k = 1, 2, \ldots, l$ then in two dimensions we define

$$\tilde{A}(i, j) = m(i - 1) + j \tag{2.11}$$

and in three dimensions

$$\tilde{A}(i, j, k) = (l \times m)(i - 1) + l(j - 1) + k. \tag{2.12}$$

NURBS basis functions for surfaces and volumes are defined by the tensor product of univariate B-spline basis functions. Let $N_{i,p}(\xi)$, $M_{j,q}(\eta)$, and $L_{l,r}(\zeta)$ be univariate B-spline basis functions associated with the knot vectors $\Xi^1 = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$, $\Xi^2 = \{\eta_1, \eta_2, \ldots, \eta_{m+q+1}\}$, and $\Xi^3 = \{\zeta_1, \zeta_2, \ldots, \zeta_{l+r+1}\}$. In two dimensions, with $A = \tilde{A}(i,j)$ and $\hat{A} = \tilde{A}(\hat{i},\hat{j})$,

$$R_A^{p,q}(\xi,\eta) = \frac{M_{i,q}(\eta)N_{j,p}(\xi)w_A}{\sum_{\hat{i}=1}^{n}\sum_{\hat{j}=1}^{m} M_{\hat{i},q}(\eta)N_{\hat{j},p}(\xi)w_{\hat{A}}} \tag{2.13}$$

are the surface NURBS basis functions. Similarly, in three dimensions, with $A = \tilde{A}(i,j,k)$ and $\hat{A} = \tilde{A}(\hat{i},\hat{j},\hat{k})$,

$$R_A^{p,q,r}(\xi,\eta,\zeta) = \frac{L_{i,r}(\zeta)M_{j,q}(\eta)N_{k,p}(\xi)w_A}{\sum_{\hat{i}=1}^{n}\sum_{\hat{j}=1}^{m}\sum_{\hat{k}=1}^{l} L_{\hat{i},r}(\zeta)M_{\hat{j},q}(\eta)N_{\hat{k},p}(\xi)w_{\hat{A}}} \tag{2.14}$$

are the volume NURBS basis functions.

Given a *control mesh* $\{\mathbf{P}_A\}$, where $A = 1, 2, \ldots, (n \times m)$ for surfaces, and $A = 1, 2, \ldots, (n \times m \times l)$ for volumes, we can define a NURBS surface as

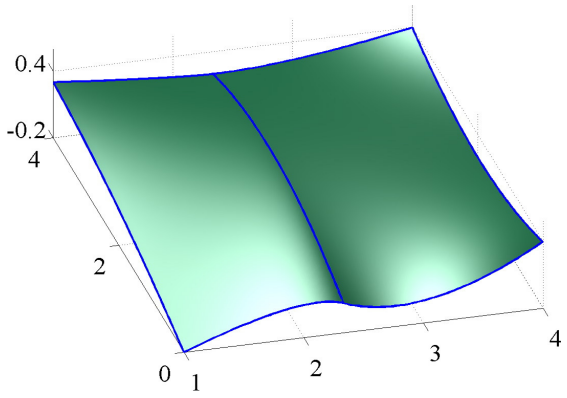
$$\mathbf{S}(\xi,\eta) = \sum_{A=1}^{n\times m} R_A^{p,q}(\xi,\eta)\mathbf{P}_A \tag{2.15}$$

and a NURBS volume as

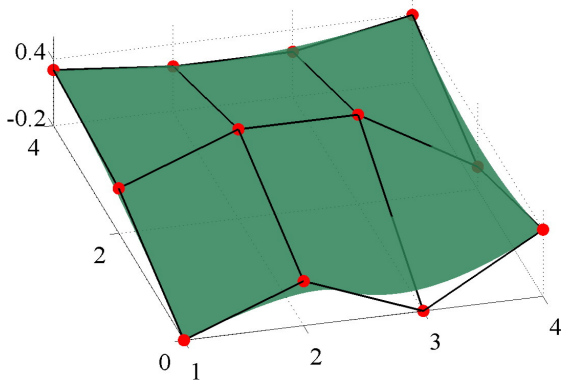
$$\mathbf{V}(\xi,\eta,\zeta) = \sum_{A=1}^{n\times m\times l} R_A^{p,q,r}(\xi,\eta,\zeta)\mathbf{P}_A. \tag{2.16}$$

2.5 The index, parameter, and physical spaces

Figure 2.3 shows both the physical mesh and control mesh in the *physical space* for a NURBS. That is, they are shown in the actual physical domain. It is also informative to consider the *parameter space*, shown in Figure 2.4, which is simply the pre-image of the physical mesh. The B-spline mapping takes each point in the parameter space to a point in the physical space, and the images of the knot lines under the NURBS mapping create the boundaries of the physical elements.



(a) Physical mesh



(b) Control mesh

Figure 2.3: A quadratic NURBS surface generated from $\Xi^1 = \{0, 0, 0, 1, 2, 2, 2\}$ and $\Xi^2 = \{0, 0, 0, 1, 1, 1\}$. a) The physical mesh is comprised of the image of the knot lines under the geometrical mapping. In this case, we have two elements. This is completely analogous to a finite element mesh. b) The control mesh is comprised of the actual control points. Note that only the corner control points lie on the surface. It is extremely important to distinguish between the physical mesh and the control mesh.

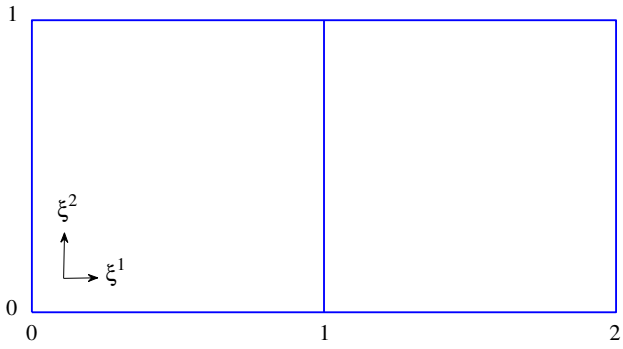


Figure 2.4: The parameter space corresponding to the biquadratic B-spline surface seen in Figure 2.3. The parameter space is the pre-image of the physical mesh.

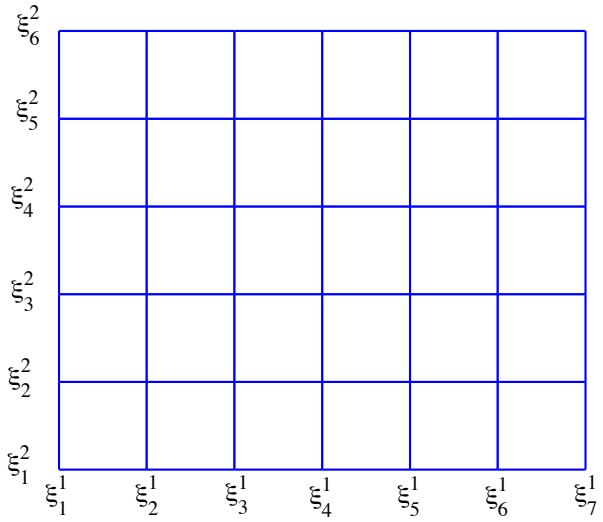


Figure 2.5: The index space corresponding to the biquadratic B-spline surface seen in Figure 2.3.

Figure 2.5 shows the *index space* for the surface. It is created by plotting the knots at equally spaced intervals, regardless of their actual values. This point of view is extremely useful for developing algorithms, as well as for building intuition about T-splines. For example, in the index space it is easy to identify the knot lines at which the support of any given function will begin or end, as well as which functions have support within any given element. This may be ambiguous in the parameter space as some of the knots may have the same value.

2.6 Knot insertion

Given a NURBS curve $\mathbf{C}(\xi)$, an equivalent NURBS curve, $\tilde{\mathbf{C}}(\xi) \equiv \mathbf{C}(\xi)$, can be created by the operation of *knot insertion*. $\tilde{\mathbf{C}}(\xi)$ is called a *refinement* of $\mathbf{C}(\xi)$. Various applications of knot insertion in the context of isogeometric analysis were discussed in [39]. Let $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ be a given knot vector. Inserting a new knot $\bar{\xi} \in [\xi_k, \xi_{k+1}[$ with $k > p$ into the knot vector requires that $n+1$ new basis functions be defined using (2.1) and (2.2) with the new knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_k, \bar{\xi}, \xi_{k+1}, \xi_{n+p+1}\}$. The $m = n+1$ new control points, $\{\bar{\mathbf{P}}_A\}_{A=1}^{m}$, are formed from the original control points, $\{\mathbf{P}_A\}_{A=1}^{n}$, by

$$\bar{\mathbf{P}}_A = \begin{cases} \mathbf{P}_1 & A = 1 \\ c_A \mathbf{P}_A + (1 - c_A)\mathbf{P}_{A-1} & 1 < A < m \\ \mathbf{P}_n & A = m \end{cases} \tag{2.17}$$

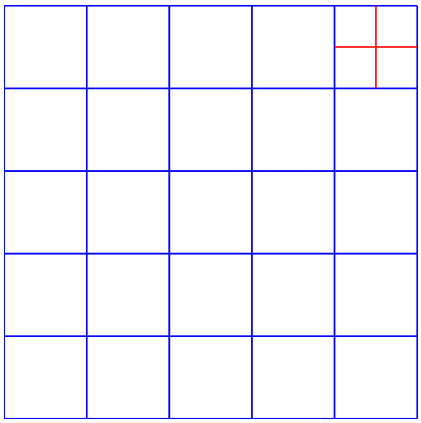
where

$$c_A = \begin{cases} 1 & 1 \leq A \leq k-p \\ \frac{\bar{\xi} - \xi_A}{\xi_{A+p} - \xi_A} & k-p+1 \leq A \leq k \\ 0 & A \geq k+1 \end{cases} \tag{2.18}$$

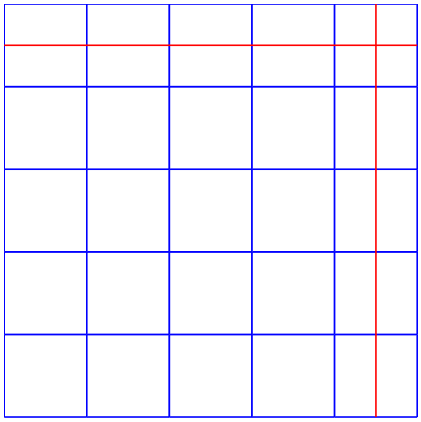
Knot values may be inserted multiple times but the continuity of the basis is reduced by one for each repetition of a given knot value. However, by choosing control variables as in (2.17) and (2.18) the continuity of the *curve* is preserved.

2.7 Limitations of a NURBS-based framework

NURBS are ubiquitous in CAD and have been successfully used as a basis for isogeometric analysis. Unfortunately, NURBS surfaces have several important drawbacks due to the fact that the surfaces are four sided. To model complicated designs requires multiple NURBS patches, which are often discontinuous across patch boundaries. Even achieving C^0 continuity across patches requires special techniques. The joining of two patches that were created separately may require the insertion of many knots and nonlinear reparameterization of one or both patches. All NURBS refinement operations are global. In other words, when we refine by inserting knots into the knot vectors of a surface, the knot lines extend throughout the entire domain (see Figure 2.6b). Global refinement is a problem in both design and analysis. To add features, such as holes, it is common to use trimming curves. The application of trimming curves destroys the analysis-suitable nature of the geometry. The geometric basis no longer describes the geometry and cannot be used directly in analysis. To achieve a tight integration of design and analysis requires a technology built on the smooth B-spline basis which can be locally refined and is capable of representing domains of arbitrary topological complexity as a single watertight geometry. All of these capabilities are present in a T-spline-based framework.



(a) Local refinement



(b) Global refinement

Figure 2.6: a) There are many instances in which we would like to locally refine an initial NURBS mesh by subdividing an individual element. b) Unfortunately, knot insertion is a global process that necessitates the propagation of the refinement throughout the domain.

Chapter 3

T-spline fundamentals

We now present fundamental concepts underlying T-spline technology. We illustrate our developments using the physical domain $\Omega \subset \mathbb{R}^2$ shown in Figure 3.1.

3.1 The T-mesh

A T-spline is constructed from a T-mesh, denoted by T. For surfaces, i.e. $d_p = 2$, the T-mesh is a mesh of quadrilateral elements[1] which permit T-junctions. In finite element parlance, a T-junction is analogous to a "hanging node." An element with T-junctions is composed of four corner vertices and any number of additional vertices on any side. The spatial representation of a T-mesh is called a T-spline control mesh. In this paper we use T-mesh and T-spline control mesh interchangeably. Every vertex in the T-mesh is assigned a control point, $\mathbf{P}_A \in \mathbb{R}^{d_s}$, and control weight, $w_A \in \mathbb{R}$, where the index A is used to denote a global control point number.

A T-mesh for the domain Ω in Figure 3.1 is shown in Figure 3.2. The black and red circles are T-mesh vertices or, equivalently, control points (see Appendix A.1 for values of the control points and weights). The T-junctions in Figure 3.2 are the red circles $\mathbf{P}_{25}$ and $\mathbf{P}_{33}$. This T-mesh will be used throughout to

[1] What we refer to as T-mesh "elements" are usually referred to as T-mesh "faces" in the CAGD literature [106].

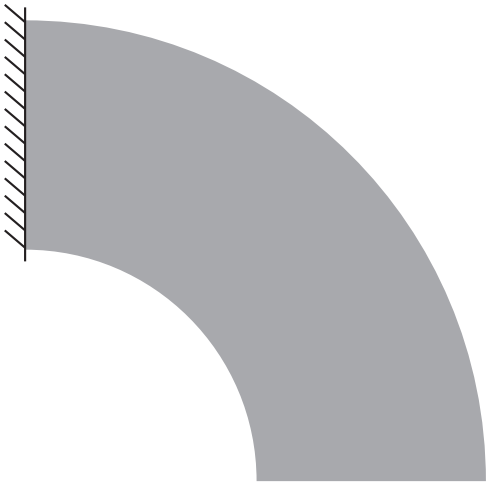


Figure 3.1: The domain $\Omega \subset \mathbb{R}^2$ for a bivariate ($d_p = 2$), cubic ($p = 3$) T-spline. The curved boundaries are exact quarter circles. The hash marks on the left indicate homogeneous Dirichlet boundary conditions.

illustrate important concepts. However, we note that this simple geometry could be represented more concisely with NURBS or T-splines. In the case of NURBS of degree 1×2, as few as six control points are capable of representing the exact geometry, and for bicubic T-splines, as few as 16 are required. The additional control points in the T-mesh of Figure 3.2 is representative of the fact that finite element analysis will typically require many more degrees of freedom than geometric design.

Each edge in a T-mesh is assigned a non-negative real number called a knot interval, which conveys the parameter distance between knot lines [108]. Knot intervals on opposite sides of every T-mesh element sum to the same value, or else the choice of knot intervals is invalid.

A valid knot interval configuration for the T-mesh in Figure 3.2 is shown in Figure 3.3. Notice that an outer ring of zero-length knot intervals has been assigned

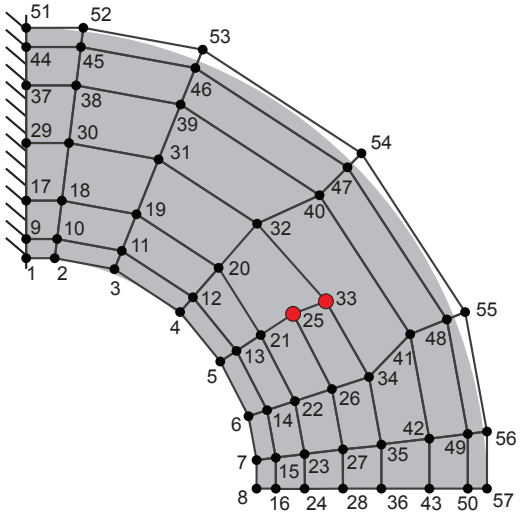


Figure 3.2: A T-mesh defining a bicubic T-spline geometry. The large red circles are the T-junctions for this T-mesh. The indexing identifies the T-mesh control points.

to the T-mesh. These zero-length knot intervals play a similar role to open knot vectors in NURBS and ease the imposition of boundary conditions.

3.2 The T-spline basis

T-spline basis functions can be inferred from a T-mesh with a valid knot interval configuration. A T-spline basis function is associated with each vertex in the T-mesh. We illustrate the construction of the T-spline basis functions associated with $\mathbf{P}_{18}$ and $\mathbf{P}_{33}$ in Figures 3.4 and 3.5, respectively.

3.2.1 Local knot interval vectors

The first step in constructing a T-spline basis function is inferring sequences of knot intervals from the T-mesh in the neighborhood of the associated vertex.

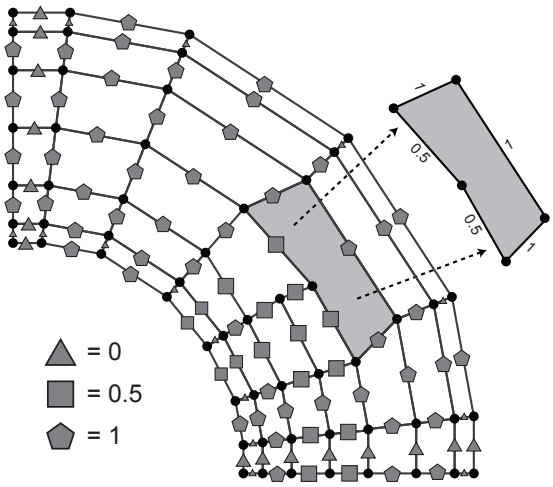


Figure 3.3: A valid knot interval configuration for the bicubic T-mesh in Figure 3.2. The triangles correspond to a knot interval of 0, the squares correspond to a knot interval of $\frac{1}{2}$, and the pentagons correspond to a knot interval of 1. A valid knot interval configuration for a T-mesh element is shown in the element callout. Notice that the knot intervals along opposing sides of the element sum to the same value.

These knot intervals are organized into local knot interval vectors. A local knot interval vector is a sequence of knot intervals, $\Delta\Xi = \{\Delta\xi_1, \Delta\xi_2, \ldots, \Delta\xi_{p+1}\}$, such that $\Delta\xi_i = \xi_{i+1} - \xi_i$, and a local knot vector, derivable from any $\Delta\Xi$, is a non-decreasing knot sequence, $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{p+2}\}$. A local knot interval vector possesses all the information in a local knot vector except an origin. For example, if a knot interval vector is $\{1, 3, 2, 1\}$, then we can set $\xi_1 = 0$ and the corresponding local knot vector is $\{0, 1, 4, 6, 7\}$. If conversely we are given a local knot vector, then the local knot interval vector is simply the difference between adjacent knots. In general, for T-splines, knot intervals are the method of choice for assigning and retrieving parameter information to and from the T-mesh since no origin is required. It should be noted that all classical B-spline algorithms can be rewritten in terms of knot intervals [103].

To every vertex, A, in the T-mesh we assign a set of local knot interval vectors, $\boldsymbol{\Delta\Xi}_A = \{\Delta\Xi_A^i\}_{i=1}^{d_p}$, from which a corresponding set of local knot vectors, $\boldsymbol{\Xi}_A = \{\Xi_A^i\}_{i=1}^{d_p}$, can be derived. The local knot interval vectors $\boldsymbol{\Delta\Xi}_A$ are constructed by marching through the T-mesh in each topological direction, starting at the the vertex A, until $p-1$ vertices or perpendicular edges are intersected. If a vertex or perpendicular edge is encountered, the knot interval distance traversed since the last intersection is placed in the local knot interval vector. If a T-mesh boundary is crossed before $p-1$ knot intervals are encountered, knot intervals are appended to complete the knot interval vector. In analysis these appended knot intervals are often chosen to be equal to zero to create an open knot vector structure along the boundary of the T-mesh. Since a knot in a local knot vector corresponds to T-mesh topology (vertices or edges) encountered during basis function inferrence the T-mesh topology and knot interval configuration determine the *knot structure* of the underlying T-spline space.

In Figure 3.4, in the upper left corner, the knot intervals used to construct the local knot vectors for T-mesh vertex $\mathbf{P}_{18}$ are shown. Notice that this vertex is near the boundary of the T-mesh. When marching to the left only a single knot interval is encountered before reaching the boundary of the T-mesh. As a result an additional zero knot interval is added to the front of the local knot interval vector. The knot interval vectors for $\mathbf{P}_{18}$ are given by

$$\boldsymbol{\Delta\Xi}_{18} = \begin{bmatrix} 0, 0, 1, 1 \\ 0, 1, 1, 1 \end{bmatrix}.$$

In Figure 3.5, in the upper left corner, the knot intervals used to construct the local knot interval vectors for T-mesh vertex $\mathbf{P}_{33}$ are shown. Notice that this is a T-junction vertex. In this case, T-mesh elements must be traversed to form the local knot interval vectors. When an element is traversed the knot interval sum associated with the sides of the element which are parallel to the traversal direction are inserted into the local knot interval vector. For $\mathbf{P}_{33}$, the knot interval vectors are given by,

$$\boldsymbol{\Delta\Xi}_{33} = \begin{bmatrix} 1, \frac{1}{2}, \frac{1}{2}, 1 \\ \frac{1}{2}, \frac{1}{2}, 1, 1 \end{bmatrix}.$$

3.2.2 The local basis function domain

Each set of local knot vectors $\boldsymbol{\Xi}_A$ defines a local basis function domain, $\widehat{\Omega}_A \subset \mathbb{R}^{d_p}$, over which a single T-spline basis function is defined. The local basis function domain is defined as

$$\widehat{\Omega}_A = \bigotimes_{i=1}^{d_p} \widehat{\Omega}_A^i, \tag{3.1}$$

where $\widehat{\Omega}_A^i = [0, \Delta\xi_1^i + \Delta\xi_2^i + \Delta\xi_3^i + \Delta\xi_4^i] \subset \mathbb{R}$. Each local basis function domain carries a coordinate system $\boldsymbol{\xi}_A = (\xi_A^1, \xi_A^2) = (\xi_A, \eta_A)$. This coordinate system is called the basis coordinate system.

Local basis function domains $\widehat{\Omega}_{18}$ and $\widehat{\Omega}_{33}$ are shown in the bottom right corners of Figures 3.4 and 3.5, respectively. In the case of $\widehat{\Omega}_{18}$ we have that $\widehat{\Omega}_{18}^1 = [0, 2]$ and $\widehat{\Omega}_{18}^2 = [0, 3]$. In the case of $\widehat{\Omega}_{33}$ we have that $\widehat{\Omega}_{33}^1 = [0, 3]$ and $\widehat{\Omega}_{33}^2 = [0, 3]$.

We note that in addition to the basis coordinate systems in a T-mesh, it is often desirable to establish larger *knot coordinate systems* for a subset of the knot structure of a T-mesh. Using a knot coordinate system, the knot vectors for all basis functions under consideration are assigned the same origin. Knot coordinate systems are used when computing the elements of the refinement operator, $\mathbf{M}$, as described in Sections 5.6.1.2 and 5.7.

3.2.3 T-spline basis functions

Over each local basis function domain $\widehat{\Omega}_A$ we define a single T-spline basis function, $N_A : \widehat{\Omega}_A \rightarrow \mathbb{R}^+ \cup 0$. This is done by forming the tensor product of the univariate basis functions $\{N_A^i(\xi_A^i \,|\, \Xi_A^i)\}_{i=1}^{d_p}$ as

$$N_A\left(\boldsymbol{\xi}_A \,|\, \boldsymbol{\Xi}_A\right) \equiv \prod_{i=1}^{d_p} N_A^i\left(\xi_A^i \,|\, \Xi_A^i\right). \tag{3.2}$$

The univariate B-spline basis function, $N_A^i : \widehat{\Omega}_A^i \rightarrow \mathbb{R}^+ \cup 0$, is defined using a recurrence relation, starting with the piecewise constant ($p = 0$) basis function

$$N_A^i\left(\xi_A^i \,|\, \xi_{A,1}^i, \xi_{A,2}^i\right) = \begin{cases} 1 & \text{if } \xi_{A,1}^i \leq \xi_A^i < \xi_{A,2}^i \\ 0 & \text{otherwise} \end{cases}, \tag{3.3}$$

where $\xi_{A,k}^i$ is the k^{th} knot value in the localized knot vector Ξ_A^i. For $p > 0$, the basis function is defined using the Cox-de Boor recursion formula:

$$\begin{aligned} N_A^i\left(\xi_A^i \,|\, \xi_{A,1}^i, \ldots, \xi_{A,p+2}^i\right) = & \frac{\xi_A^i - \xi_{A,1}^i}{\xi_{A,p+1}^i - \xi_{A,1}^i} N_A^i\left(\xi_A^i \,|\, \xi_{A,1}^i, \ldots, \xi_{A,p+1}^i\right) + \\ & + \frac{\xi_{A,p+2}^i - \xi_A^i}{\xi_{A,p+2}^i - \xi_{A,2}^i} N_A^i\left(\xi_A^i \,|\, \xi_{A,2}^i, \ldots, \xi_{A,p+2}^i\right). \end{aligned} \tag{3.4}$$

The T-spline basis functions N_{18} and N_{33} are shown in the lower left corner of Figures 3.4 and 3.5, respectively. It should be noted that while algorithms exist for computing T-spline basis functions according to (3.4) (see [88]), using the recursive definition in finite element shape function routines is expensive when compared to the extracted element technology introduced in Section 4. Between knots, the one-dimensional basis functions (3.4) are C^{∞} continuous. At knots the continuity is reduced (see [38]).

3.3 T-spline volumes

The extension of T-splines to the trivariate setting is relatively straightforward. As an example, consider the volumetric T-mesh shown below in Figure 3.6.

Analogous to the surface case, a volumetric T-mesh consists of a set of hexahedral elements which permit T-junctions. Now, each quadrilateral face of a T-mesh element corresponds to a knot value. The inferrence of local knot vectors, and in turn, basis functions, is similar to the bivariate case. Whereas in the bivariate case we only recorded knots when we encountered T-mesh vertices or perpendicular edges, now we also record a knot whenever we encounter a perpendicular quadrilateral face. This is illustrated in Figure 3.7.

Once we determine the local knot vectors, the set of T-spline basis functions are constructed in exactly the same manner as described in Section 3.2.

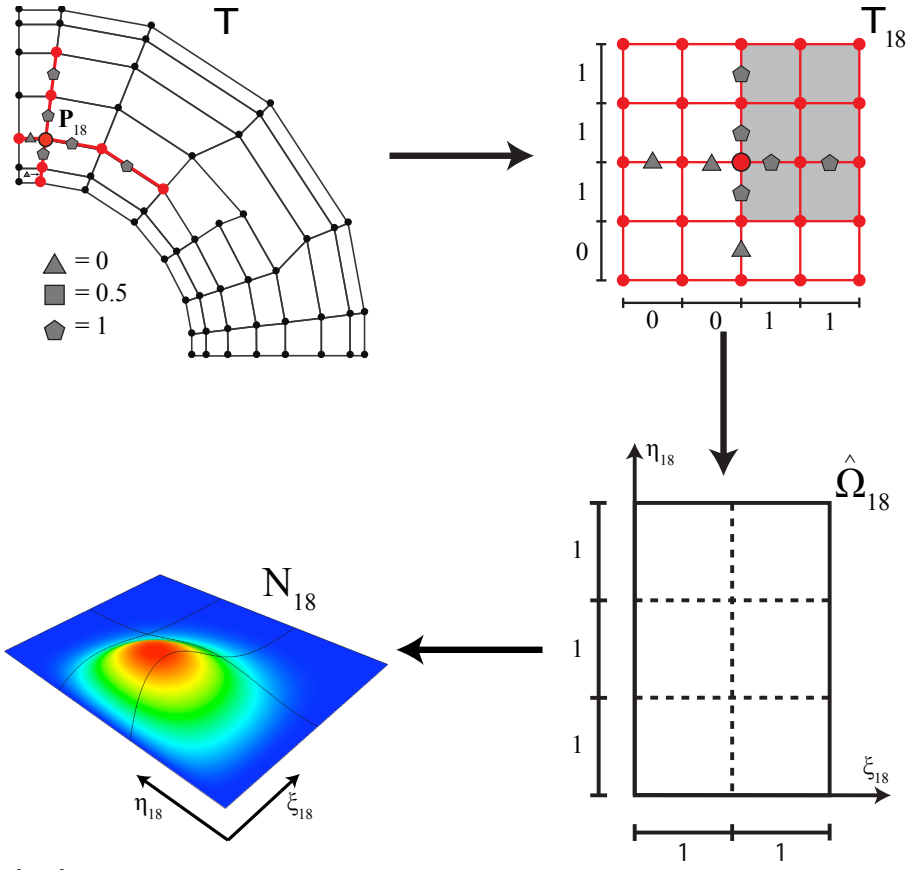


Figure 3.4: The construction of T-spline basis function N_{18} which is associated with T-mesh vertex $\mathbf{P}_{18}$. Starting at the upper left and going clockwise we have: Inference of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function.

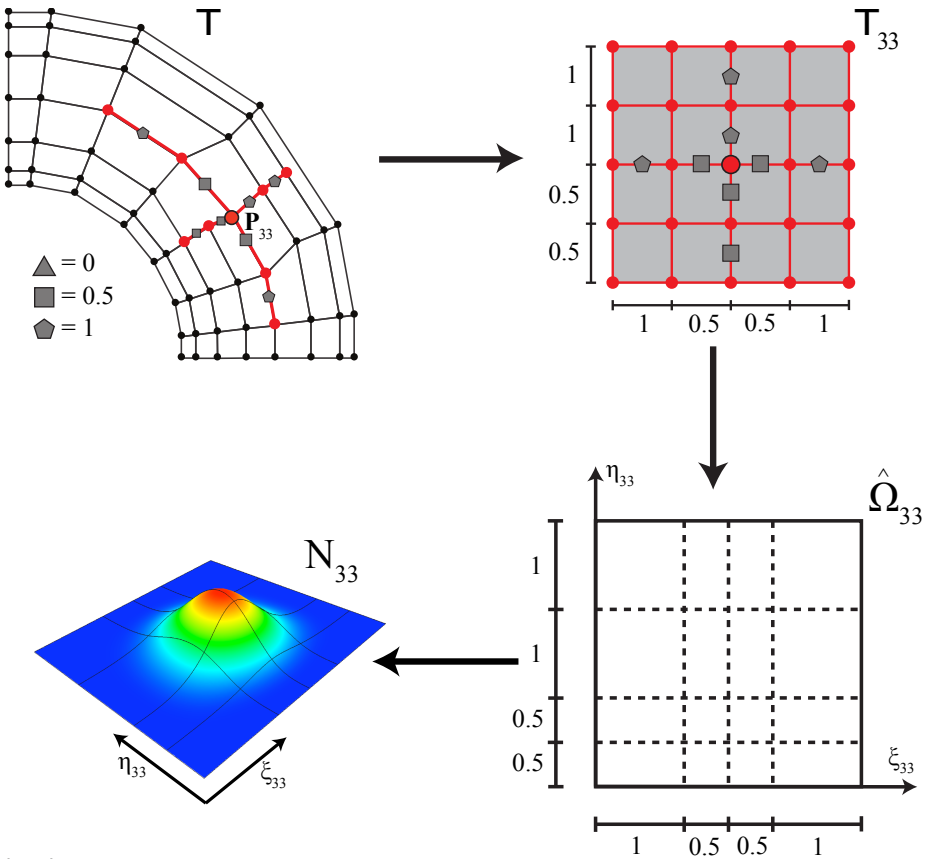


Figure 3.5: The construction of T-spline basis function N_{33} which is associated with the T-junction T-mesh vertex $\mathbf{P}_{33}$. Starting at the upper left and going clockwise we have: Inferrence of the local knot vectors from the T-mesh, the resulting local basis function mesh, the local basis function domain, and the T-spline basis function.

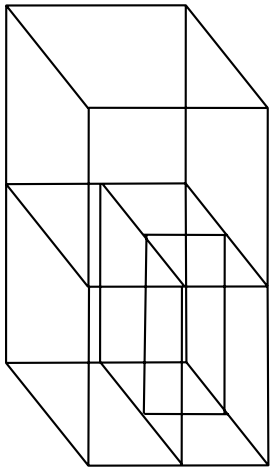


Figure 3.6: A volumetric T-mesh

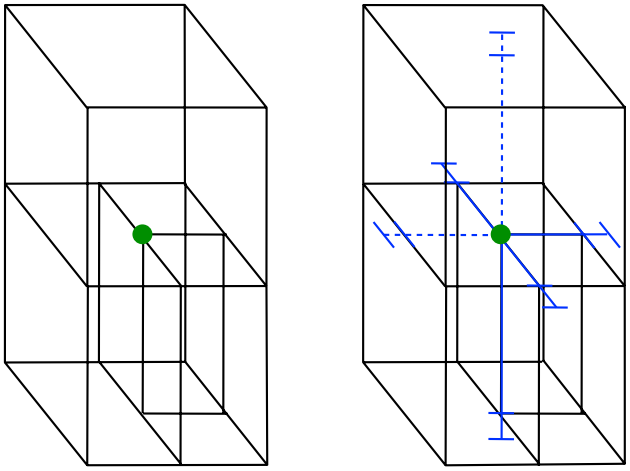


Figure 3.7: Extracting a local knot vector from a volumetric T-mesh

3.4 The T-spline element structure

We now develop the finite element structure underlying T-splines. A T-spline element $\Omega^e \subset \mathbb{R}^{d_s}$ is a region in physical space which is bounded by knot lines, which are lines of reduced continuity in the T-spline basis. We use the terminologies knot lines and lines of reduced continuity interchangeably. The basis functions restricted to the interior of the T-spline element are C^∞.

3.4.1 The local basis function mesh

From any set of local knot interval vectors $\boldsymbol{\Delta\Xi}_A$ (and corresponding local knot vectors $\boldsymbol{\Xi}_A$) a local basis function mesh, T_A, can be defined as the tensor product mesh representation of the local knot vectors

$$\mathsf{T}_A = \bigotimes_{i=1}^{d_p} \Xi_A^i. \tag{3.5}$$

Each edge in T_A continues to carry the appropriate knot interval from $\boldsymbol{\Delta\Xi}_A$.

Figure 3.4, in the upper right corner, shows the local basis function mesh, T_{18}, generated from $\boldsymbol{\Delta\Xi}_{18}$. The shaded region indicates the local basis function mesh elements with positive parametric area. Figure 3.5, in the upper right corner, shows the local basis function mesh, T_{33}, generated from $\boldsymbol{\Delta\Xi}_{33}$. In this case every element has positive parametric area.

3.4.2 The elemental T-mesh

In general, there is not a one-to-one correspondence between the set of T-mesh elements and the set of T-spline elements. We recall that a T-mesh element is a quadrilateral in the T-mesh or T-spline control mesh and the T-spline element is a region of the T-spline surface bounded by lines of reduced continuity in the T-spline

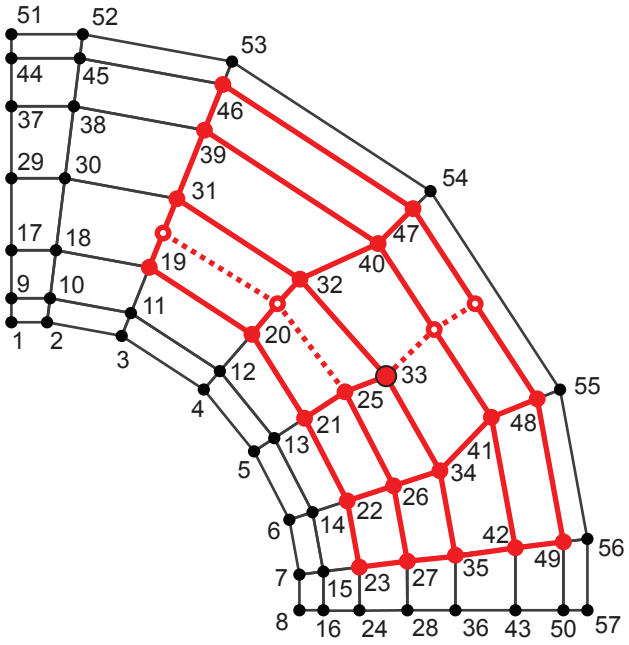


Figure 3.8: The mapping of T_{33} onto the global T-mesh T. The dashed edges correspond to lines of reduced continuity from T_{33} which are not in T.

basis. This can be seen by drawing the local basis function mesh T_{33} on top of the T-mesh, as in Figure 3.8. The dashed lines in Figure 3.8 indicate edges which exist in T_{33} but not in T. Each knot line in the T-spline basis is present in at least one basis function. However, not all these knot lines are represented by corresponding lines in the T-mesh. We form the ***elemental T-mesh*** by augmenting the T-mesh with all these lines. The elemental T-mesh T_{elem} of T is shown in Figure 3.9. The dashed edges indicate lines of reduced continuity which were not present in the original T-mesh.

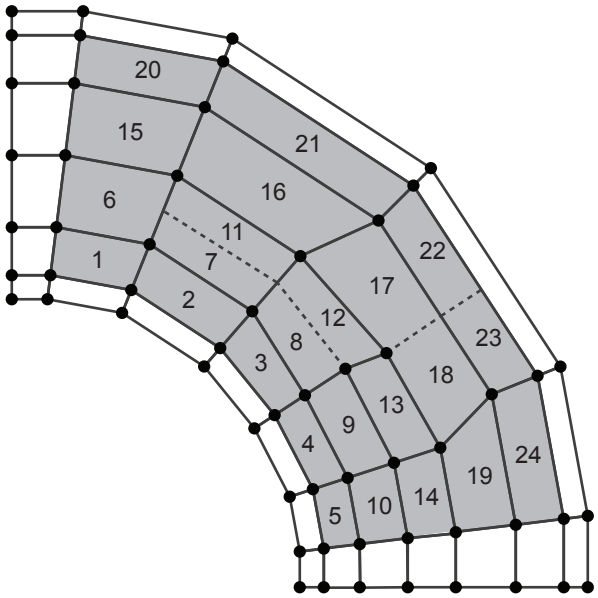


Figure 3.9: The T-spline elements in the elemental T-mesh T_f of T.

3.4.3 The IEN array

Over every element domain there exists a set of T-spline basis functions which are non-zero. The T-spline basis functions are in one-to-one correspondence with the T-mesh control points and are indexed by the global control point numbers. The IEN array maps the local basis function number, a, and the element number, e, to the corresponding global control point number A. There can be different numbers of T-spline basis functions supported by each element. This is in contrast to NURBS where all elements are in the support of exactly $(p+1)^{d_p}$ NURBS basis functions. The complete IEN array for the T-spline elements in Figure 3.9 is shown in Table 3.1. Notice that elements 9 and 10 are in the supports of 17 T-spline basis functions. Figure 3.10 shows T-spline elements 1, 10, 11, and 17 and the T-mesh control points whose corresponding basis functions are non-zero over each of these elements.

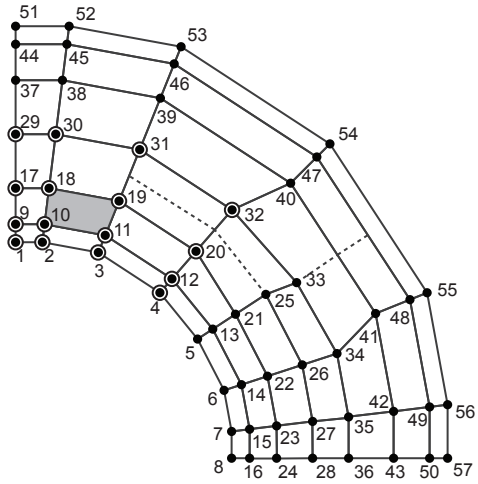
3.4.4 Restricting the global basis to the parent element domain

We develop the finite element point-of-view for T-splines by defining the T-spline basis functions over the parent element domain. The non-local and T-junction structure of the T-spline basis requires additional machinery not found in traditional finite element constructions. We define a set, $\mathbf{S}^e = \{\tilde{\Phi}^e, \{\hat{\Phi}^e_a\}_{a=1}^{n_e}\}$, of affine mappings for the element under consideration where

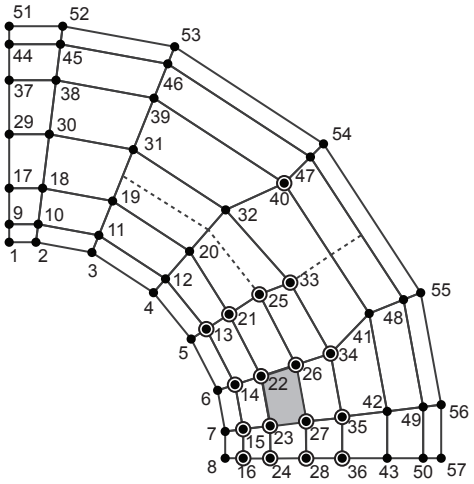
- $\tilde{\Phi}^e : \tilde{\Omega} \rightarrow \widehat{\Omega}^e$ is a one-to-one and onto affine map from the parent element domain onto the element domain:

$$\boldsymbol{\xi}^e = \tilde{\Phi}^e(\tilde{\boldsymbol{\xi}}). \tag{3.6}$$

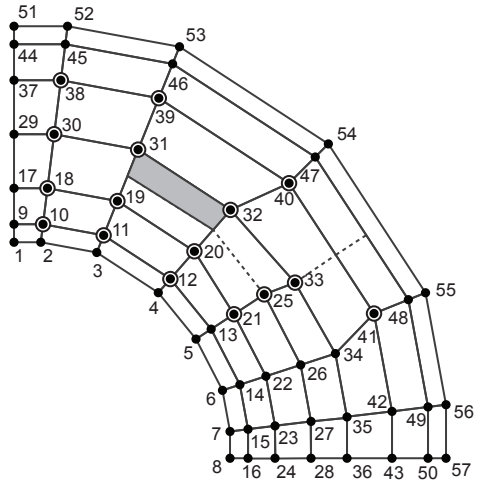
- $\hat{\Phi}^e_a : \widehat{\Omega}^e \rightarrow \widehat{\Omega}_A$, for $a = 1, \ldots, n_e$ is a one-to-one affine mapping from the element domain into the local basis function domain for basis function



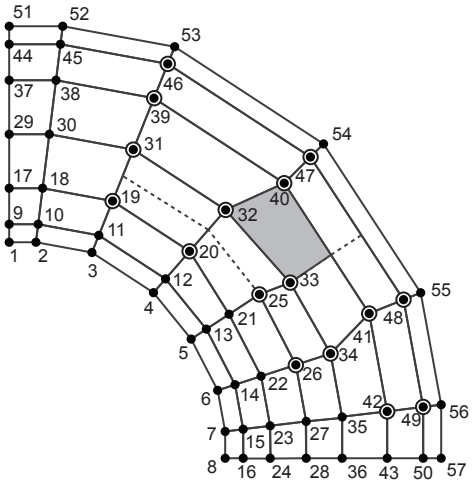
T-mesh element 1



T-mesh element 10



T-mesh element 11



T-mesh element 17

◉ Indicates T-spline basis functions that are supported by the highlighted element

Figure 3.10: T-mesh elements 1, 10, 11, and 17 in the elemental T-mesh and the T-mesh control points whose corresponding T-spline basis functions are non-zero over these T-spline elements.

	Element function number (a)																
e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	9	10	11	12	17	18	19	20	29	30	31	32	
2	2	3	4	5	10	11	12	13	18	19	20	21	25	30	31	32	
3	3	4	5	6	11	12	13	14	19	20	21	22	25	26	31	32	
4	4	5	6	7	12	13	14	15	20	21	22	23	25	26	27	32	
5	5	6	7	8	13	14	15	16	21	22	23	24	25	26	27	28	
6	9	10	11	12	17	18	19	20	29	30	31	32	37	38	39	40	
7	10	11	12	13	18	19	20	21	25	30	31	32	33	38	39	40	
8	11	12	13	14	19	20	21	22	25	26	31	32	33	34	39	40	
9	12	13	14	15	20	21	22	23	25	26	27	32	33	34	35	39	40
10	13	14	15	16	21	22	23	24	25	26	27	28	33	34	35	36	40
11	10	11	12	18	19	20	21	25	30	31	32	33	38	39	40	41	
12	11	12	19	20	21	22	25	26	31	32	33	34	39	40	41	42	
13	12	20	21	22	23	25	26	27	32	33	34	35	39	40	41	42	
14	21	22	23	24	25	26	27	28	33	34	35	36	40	41	42	43	
15	17	18	19	20	29	30	31	32	37	38	39	40	44	45	46	47	
16	18	19	20	25	30	31	32	33	38	39	40	41	45	46	47	48	
17	19	20	25	26	31	32	33	34	39	40	41	42	46	47	48	49	
18	20	25	26	27	32	33	34	35	39	40	41	42	46	47	48	49	
19	25	26	27	28	33	34	35	36	40	41	42	43	47	48	49	50	
20	29	30	31	32	37	38	39	40	44	45	46	47	51	52	53	54	
21	30	31	32	33	38	39	40	41	45	46	47	48	52	53	54	55	
22	31	32	33	34	39	40	41	42	46	47	48	49	53	54	55	56	
23	32	33	34	35	39	40	41	42	46	47	48	49	53	54	55	56	
24	33	34	35	36	40	41	42	43	47	48	49	50	54	55	56	57	

$$A = \mathrm{IEN}(a, e)$$

Table 3.1: The IEN array is constructed using the information from Figure 3.10. The IEN array maps the local basis function number (a) and the element number (e) to the corresponding global control point (A). The local basis function number indexes the T-spline basis functions supported by the element in question. Note that there can be different numbers of T-spline basis functions associated with different elements.

$A = \mathrm{IEN}(a, e)$:

$$\boldsymbol{\xi}_A = \hat{\Phi}^e_a(\boldsymbol{\xi}^e). \tag{3.7}$$

These mappings can be determined using the elemental T-mesh and the IEN array and are used to map from the parent element into each local basis function domain which corresponds to a T-spline basis function which is non-zero over element e. The action of some but not all of the affine maps in $\mathbf{S}^{17}$ is shown in Figure 3.11.

Using $\mathbf{S}^e$ and the IEN array we can define a localized, element-based definition for the global T-spline basis functions as

$$N_A(\boldsymbol{\xi}_A)|_e = N_A\left(\hat{\Phi}^e_a(\boldsymbol{\xi}^{\boldsymbol{e}})\right)\Big|_e = N_A\left(\hat{\Phi}^e_a\left(\tilde{\Phi}^e(\tilde{\boldsymbol{\xi}})\right)\right)\Big|_e = N^e_a(\tilde{\boldsymbol{\xi}}) \tag{3.8}$$

where $|_e$ indicates restriction to the domain of element number e.

3.4.5 The T-spline element geometric map

With the basis functions defined over the parent element by equation (3.8), we now define the element geometric map, $\mathbf{x}^e : \tilde{\Omega} \to \Omega^e$, from the parent element domain onto the physical domain as

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \frac{\sum_{a=1}^{n_e} \mathbf{P}^e_a w^e_a N^e_a(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})} = \sum_{a=1}^{n_e} \mathbf{P}^e_a R^e_a(\tilde{\boldsymbol{\xi}}) \tag{3.9}$$

where

$$W^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_e} w^e_a N^e_a(\tilde{\boldsymbol{\xi}}) \tag{3.10}$$

is the element weight function, and $\mathbf{P}^e_a = \mathbf{P}_{\mathrm{IEN(a,e)}}$ and $w^e_a = w_{\mathrm{IEN(a,e)}}$ are the control point and weight, respectively, corresponding to the a^{th} T-spline basis function over element e. R^e_a is the rational form of the T-spline basis function because it includes the weight function in its denominator. Defining the element weight vector

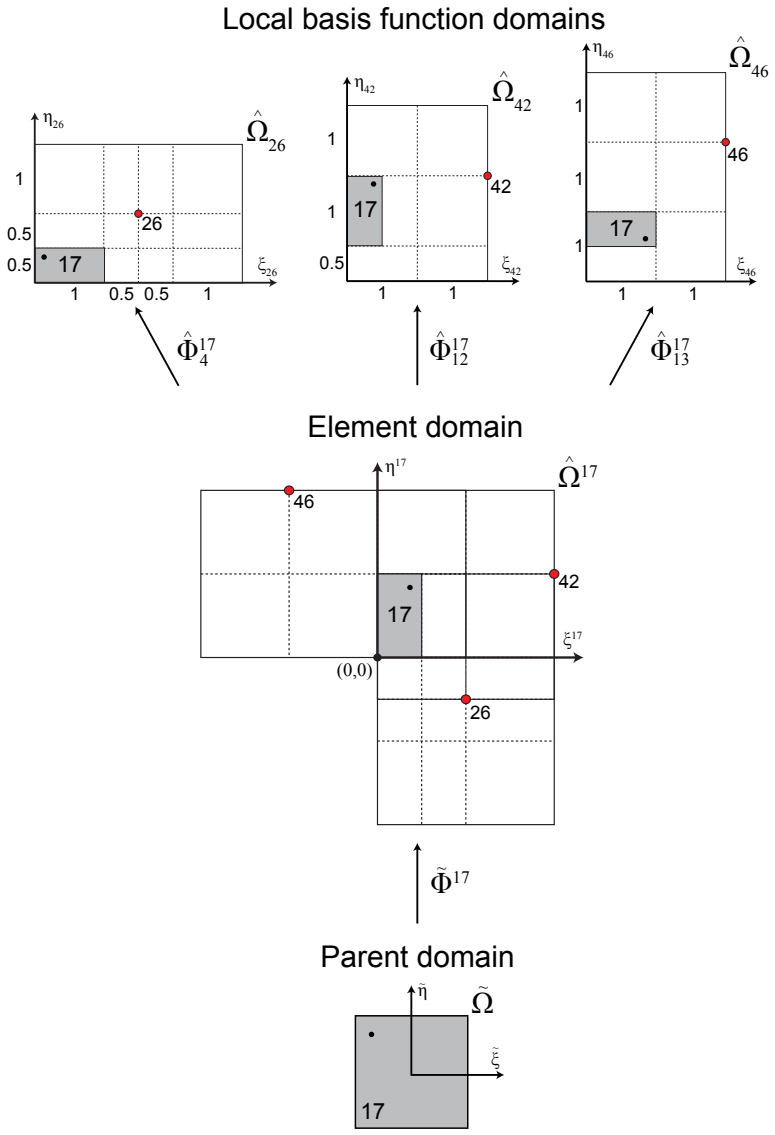


Figure 3.11: The mappings between parent, element, and basis function coordinate systems for element 17. For simplicity only some of the mappings are shown. $\tilde{\Phi}^{17}$ maps from the parent coordinate system into the element coordinate system. $\hat{\Phi}_a^{17}$ maps from the element coordinate system of local basis function number a into the basis function coordinate system of global control point $A = \mathrm{IEN}(a, 17)$. The solid dot $(\bullet)$ on element 17 shows the rotations incorporated in the mappings.

$\mathbf{w}^e = \{w_a^e\}_{a=1}^{n_e}$, the diagonal weight matrix $\mathbf{W}^e = \mathrm{diag}(\mathbf{w}^e)$, and element control points $\mathbf{P}^e$ as a matrix of dimension $n_e \times d_s$,

$$\mathbf{P}^e = \begin{bmatrix} P_1^{e,1} & P_1^{e,2} & \dots & P_1^{e,d_s} \\ P_2^{e,1} & P_2^{e,2} & \dots & P_2^{e,d_s} \\ \vdots & \vdots & & \vdots \\ P_{n_e}^{e,1} & P_{n_e}^{e,2} & \dots & P_{n_e}^{e,d_s} \end{bmatrix}, \tag{3.11}$$

we can generate the corresponding matrix representation of the geometric map as

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \frac{1}{(\mathbf{w}^e)^T \mathbf{N}^e(\tilde{\boldsymbol{\xi}})} (\mathbf{P}^e)^T \mathbf{W}^e \mathbf{N}^e(\tilde{\boldsymbol{\xi}}) \tag{3.12}$$

$$= (\mathbf{P}^e)^T \mathbf{R}^e(\tilde{\boldsymbol{\xi}}), \tag{3.13}$$

where $\mathbf{N}^e = \{N_a^e\}_{a=1}^{n_e}$ and $\mathbf{R}^e = \{R_a^e\}_{a=1}^{n_e}$ are the vectors of polynomial and rational T-spline basis functions, respectively, which are non-zero over element e.

Chapter 4

Bézier extraction

The T-spline element construction presented in Section 3.4 can be simplified by expressing the elements in Bézier form. We call this *Bézier extraction*. Bézier extraction for T-splines was briefly mentioned in the context of geometric design in [106]. The application of Bézier extraction to isogeometric analysis for the special case of NURBS was detailed in [23]. The idea is to extract the linear operator which maps the Bernstein polynomial basis on Bézier elements to the global T-spline basis. This operator is a simple, compact, algebraic representation of the topological and smoothness information stored in the elemental T-mesh and T-spline basis. The linear transformation is defined by a matrix referred to as the extraction operator. The transpose of the extraction operator maps the control points of the global T-spline to the control points of the Bernstein polynomials. Figure 4.1 illustrates the idea for a B-spline curve. This provides a finite element representation of T-splines, and facilitates the incorporation of T-splines into existing finite element programs. Only the shape function subroutine needs to be modified. All other aspects of the finite element program remain the same. Additionally, Bézier extraction is automatic and can be applied to any T-spline regardless of topological complexity or polynomial degree. In particular, it represents an elegant treatment of T-junctions, referred to as "hanging nodes" in finite element analysis.

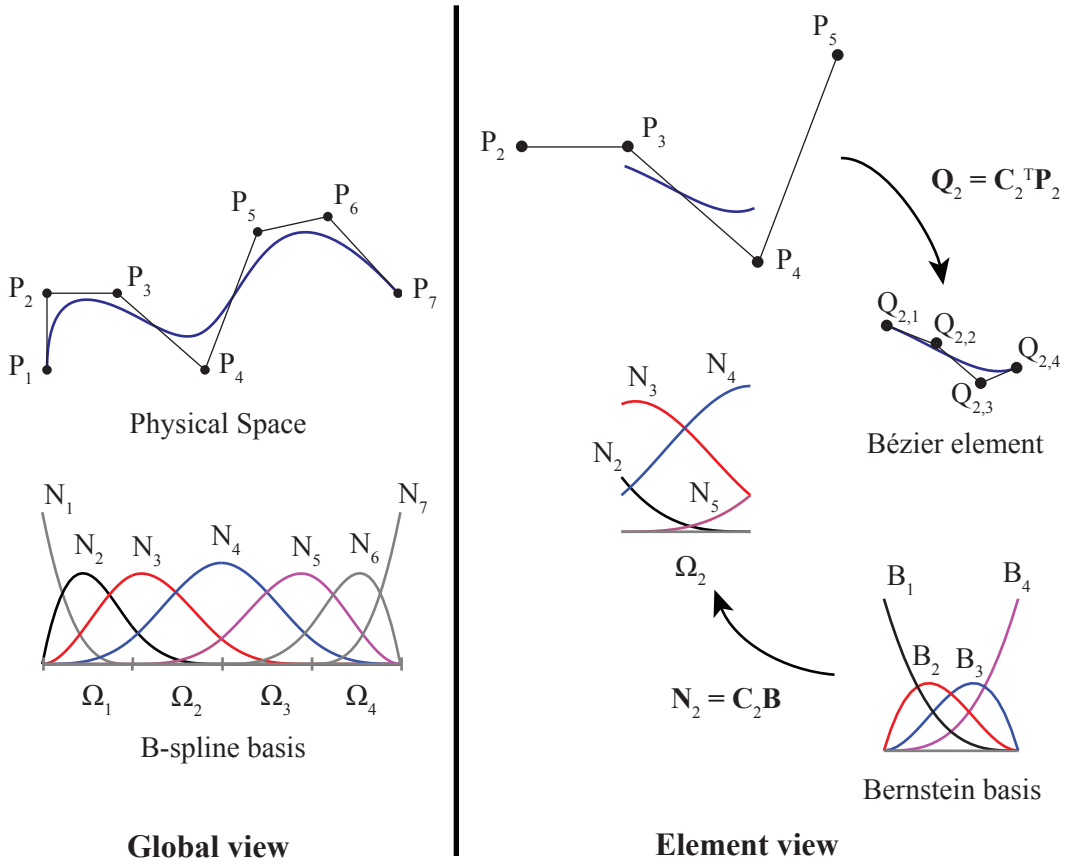


Figure 4.1: Bézier extraction for a cubic B-spline curve. The B-spline curve and basis functions are shown on the left. The action of the extraction operator, $\mathbf{C_2}$, for element Ω_2 is illustrated on the right. The transpose of the extraction operator defines the control points, $\mathbf{Q}_2 = \{\mathrm{Q}_{2,I}\}_{I=1}^{4}$, of the Bézier element from the control points, $\mathbf{P}_2 = \{\mathrm{P}_I\}_{I=2}^{5}$ of the B-spline curve. The B-spline basis functions, $\mathbf{N}_2 = \{\mathrm{N}_I\}_{I=2}^{5}$, can be computed over the element by applying the extraction operator to Bernstein polynomial basis functions, $\mathbf{B} = \{\mathrm{B}_I\}_{I=1}^{4}$, defined on the Bézier element. Note that the Bernstein basis is the same for each element. Formation of element arrays can thus be standardized in a shape function routine.

4.1 The element extraction operator and the Bézier element

Bézier extraction for T-splines determines the exact representation of the T-spline basis over each T-spline element e in terms of a set of Bernstein polynomials,

$\mathbf{B}(\tilde{\boldsymbol{\xi}})$. Every localized T-spline basis function, $N_a^e(\tilde{\boldsymbol{\xi}})$, can be written as a linear combination of these Bernstein polynomials. In other words, for each localized T-spline basis function, $N_a^e(\tilde{\boldsymbol{\xi}})$, there exists coefficients, $c_{a,b}^e$ such that [1]

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \sum_{b=1}^{(p+1)^{d_p}} c_{a,b}^e B_b(\tilde{\boldsymbol{\xi}}) \tag{4.1}$$

over element e. In matrix-vector form (4.1) is written as

$$\mathbf{N}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{C}^e \mathbf{B}(\tilde{\boldsymbol{\xi}}), \tag{4.2}$$

where $\mathbf{C}^e$ is the element extraction operator. We call the element defined by the Bernstein polynomials the Bézier element.

Note that, in contrast with the T-spline basis functions $\mathbf{N}^e$ and $\mathbf{R}^e$, there are the same number of Bernstein basis functions for all the elements. Also, the use of the operator allows us to standardize the form of the element basis on the parent domain. In other words, each Bézier element is defined in terms of the exact same set of Bernstein basis functions. This may be contrasted with the T-spline basis defined over each T-spline element in which the structure of the basis changes from element to element.

4.2 The Bernstein basis

The Bernstein polynomials form a basis for the Bézier element. The univariate Bernstein basis functions are defined over the biunit interval $[-1, 1]$ as

$$B_{i,p}^k(\tilde{\xi}^k) = \frac{1}{2^p} \binom{p}{i-1} (1 - \tilde{\xi}^k)^{p-(i-1)} (1 + \tilde{\xi}^k)^{i-1} \tag{4.3}$$

[1] Note that the element basis functions are numbered $b = 1, 2, \ldots, (p+1)^{d_p}$. This convention is typical in finite element analysis, but different than that used in CAGD in which it is standard for the indexing to begin with 0.

where the binomial coefficient $\binom{p}{i-1} = \frac{p!}{(i-1)!(p+1-i)!}$, $1 \leq i \leq p+1$. In CAGD, the Bernstein polynomials are usually defined over the unit interval $[0, 1]$, but in finite element analysis the biunit interval is preferred to take advantage of the usual domains for Gauss quadrature. The univariate Bernstein basis functions for $p = 1, 2$, and 3 are plotted in Figure 4.2. The univariate Bernstein basis has the following properties:

- Partition of unity.

$$\sum_{i=1}^{p+1} B_{i,p}^k(\tilde{\xi}^k) = 1 \quad \forall \tilde{\xi}^k \in [-1, 1] \tag{4.4}$$

- Pointwise nonnegativity.

$$B_{i,p}^k(\tilde{\xi}^k) \geq 0 \quad \forall \tilde{\xi}^k \in [-1, 1] \tag{4.5}$$

- Endpoint interpolation.

$$B_{1,p}^k(-1) = B_{p+1,p}^k(1) = 1 \tag{4.6}$$

- Symmetry.

$$B_{i,p}^k(\tilde{\xi}^k) \equiv B_{p+1-i,p}^k(-\tilde{\xi}^k) \tag{4.7}$$

The multivariate Bernstein basis functions of degree p, $B_{a,p} : \tilde{\Omega} \to \mathbb{R}^+ \cup 0$, with $a = 1, \ldots, (p+1)^{d_p}$, are formed as the tensor-product of univariate basis functions $B_{i,p}^k : [-1, 1] \to \mathbb{R}^+ \cup 0$, with $i = 1, \ldots, p+1$. In the bivariate case we have

$$B_{a(i,j),p}(\tilde{\boldsymbol{\xi}}) = B_{i,p}^1(\tilde{\xi}^1) B_{j,p}^2(\tilde{\xi}^2). \tag{4.8}$$

with

$$a\,(i, j) = (p+1)(j-1) + i. \tag{4.9}$$

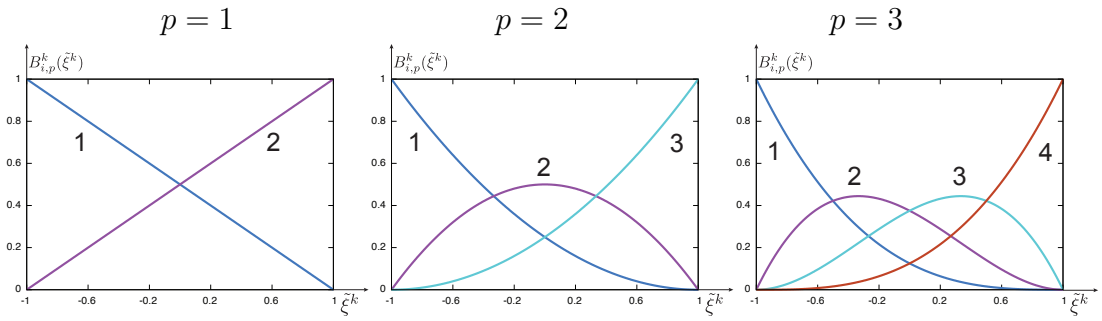


Figure 4.2: Bernstein basis functions for polynomial degree $p = 1, 2, 3$.

Recall that $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}^1, \tilde{\xi}^2) = (\tilde{\xi}, \tilde{\eta})$ is the coordinate system assigned to the parent element. From the formulas, it is clear that the basis functions are numbered from left to right in one dimension. In two dimensions each row is numbered from left to right, starting with the bottom row and moving upward. See Figure 4.3.

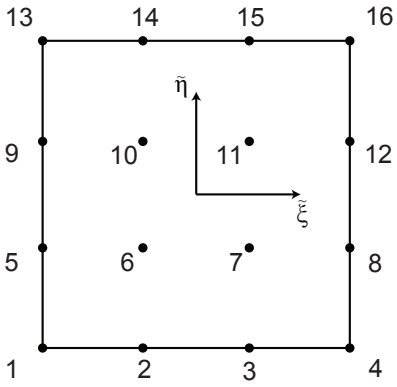


Figure 4.3: Ordering of the two-dimensional Bernstein polynomials on a bicubic Bézier element.

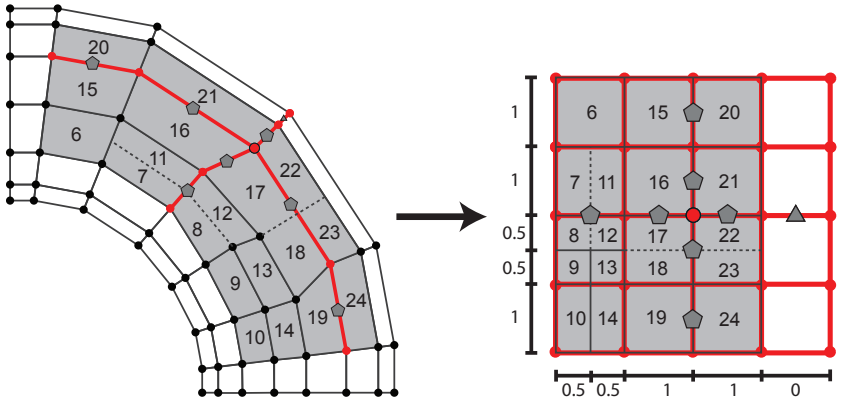


Figure 4.4: All Bézier elements in the support of T-spline basis function N_{40} in the elemental T-mesh, T_f (left) and the local basis function mesh, T_{40} (right).

4.3 Computing the element extraction operator for T-splines

T-splines that have extraordinary points (see Chapter 6) do not have a global rectangular parameter domain but, as was discussed in Section 3.2.2, a local rectangular parameter domain can be defined for each individual basis function. Thus, for T-splines, the computation of the element extraction operators is performed function-by-function, where each basis function contributes a row to each of the extraction operators corresponding to the Bézier elements in its support. For example, Figure 4.4 shows the Bézier elements in the support of T-spline basis function N_{40}. Bézier extraction of N_{40} contributes a row to the element extraction operator for each of the elements in its support.

4.3.1 Univariate extraction

We now describe Bezier extraction for univariate B-spline basis functions, which means that we represent each B-spline basis function in terms of $p+1$ polyno-

mials in Bernstein form. The extraction operator rows are computed by performing repeated knot insertion at all knots in a local knot vector and any requested interior knot locations so that every knot has multiplicity p.

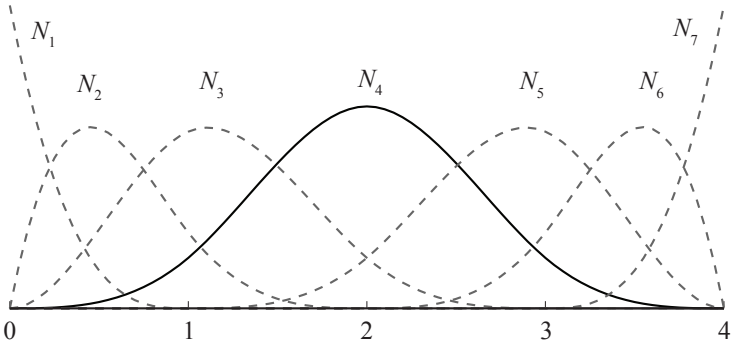


Figure 4.5: A univariate B-spline basis function, N_4, (solid line) with local knot vector $\Xi = \{0, 1, 2, 3, 4\}$.

For example, repeated knot insertion at each existing knot location for the local knot vector, $\Xi = \{0, 1, 2, 3, 4\}$, corresponding to B-spline basis function N_4 in Figure 4.5 produces the Bernstein basis functions shown in Figure 4.6. Table 4.1 shows the full extraction operators corresponding to the four univariate Bézier elements in Figure 4.5 where the rows corresponding to N_4 are highlighted. The extraction operator rows are computed by Algorithm 1, which is a modified version of the algorithm presented for NURBS in [23].

As a result of the extraction process we have a representation of each B-spline function in terms of the basis functions of the Bézier elements. This can be seen by considering Bézier element e in the support of the B-spline basis function A. The B-spline basis function A restricted to element e is related to the Bernstein basis functions by

$$N_A(\boldsymbol{\xi}_A)|_e = N_a^e(\tilde{\boldsymbol{\xi}}) = \mathbf{e}_a^T \mathbf{N}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{e}_a^T \mathbf{C}^e \mathbf{B}(\tilde{\boldsymbol{\xi}}) = \mathbf{c}_a^{e\,T} \mathbf{B}(\tilde{\boldsymbol{\xi}}) \tag{4.10}$$

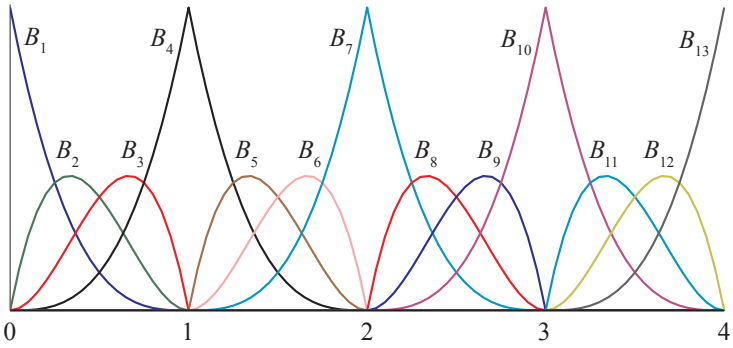


Figure 4.6: The resulting basis functions of the Bézier elements after knot insertion and refinement of the basis functions shown in Figure 4.5.

$$\begin{Bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1/2 & 1/4 \\ 0 & 0 & 1/2 & 7/12 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{Bmatrix}, \quad \begin{Bmatrix} N_2 \\ N_3 \\ N_4 \\ N_5 \end{Bmatrix} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 7/12 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \begin{Bmatrix} B_4 \\ B_5 \\ B_6 \\ B_7 \end{Bmatrix},$$

$$\begin{Bmatrix} N_3 \\ N_4 \\ N_5 \\ N_6 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 2/3 & 2/3 & 1/3 & 1/6 \\ 1/6 & 1/3 & 2/3 & 7/12 \\ 0 & 0 & 0 & 1/4 \end{bmatrix} \begin{Bmatrix} B_7 \\ B_8 \\ B_9 \\ B_{10} \end{Bmatrix}, \quad \begin{Bmatrix} N_4 \\ N_5 \\ N_6 \\ N_7 \end{Bmatrix} = \begin{bmatrix} 1/6 & 0 & 0 & 0 \\ 7/12 & 1/2 & 0 & 0 \\ 1/4 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} B_{10} \\ B_{11} \\ B_{12} \\ B_{13} \end{Bmatrix}$$

Table 4.1: The extraction operators corresponding to the four Bézier elements in Figure 4.5. The highlighted rows correspond to the extraction of N_4 over each element.

where $\mathbf{e}_a$ is a unit vector which is equal to 1 in entry a and zero elsewhere. The vector $\mathbf{c}_a^e$ extracts basis function $A = \mathrm{IEN}(a, e)$ from Bézier element e.

4.3.2 Multivariate extraction

Multivariate Bézier extraction operators $\mathbf{C}^e$ for a T-spline element e are constructed as products of univariate Bézier extraction operators. In the bivariate case,

T-spline basis function N_A restricted to element e can be decomposed into two univariate B-spline basis functions as

$$N_A(\boldsymbol{\xi}_A)|_e = N_a^e(\tilde{\boldsymbol{\xi}}) = N_a^{e,1}(\tilde{\xi}^1)N_a^{e,2}(\tilde{\xi}^2), \tag{4.11}$$

where a and e are such that $A = \mathrm{IEN}(a, e)$ and the superscripts indicate the local parametric direction. Using the extraction procedure outlined above for univariate B-spline basis functions, we can construct extraction operators for the basis functions $N_a^{e,1}(\tilde{\xi}^1)$ and $N_a^{e,2}(\tilde{\xi}^2)$. By substituting equation (4.10) into (4.11) we obtain

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \left[\mathbf{c}_a^{e,1\,T}\,\mathbf{B}^1(\tilde{\xi}^1)\right]\left[\mathbf{c}_a^{e,2\,T}\,\mathbf{B}^2(\tilde{\xi}^2)\right]. \tag{4.12}$$

Using the index mapping from equation (4.9) it can be shown that this becomes

$$N_a^e(\tilde{\boldsymbol{\xi}}) = \mathbf{c}_a^{e\,T}\,\mathbf{B}(\tilde{\boldsymbol{\xi}}), \tag{4.13}$$

where now $\mathbf{c}_a^e$ corresponds to the a^{th} row of the bivariate element extraction operator $\mathbf{C}^e$ in equation (4.2). This process is repeated for each T-spline basis function supported by element e to generate the full extraction operator for the element. Importantly, once the element extraction operators are computed, a finite element code never needs to know anything about T-mesh topology to process the global T-spline basis.

The process and result of extracting $N_{40}(\boldsymbol{\xi}_{40})$ over an element e in its support are shown in Figures 4.7 and 4.8, respectively. In Figure 4.7 the univariate components of the T-spline basis function are shown on the left and right. Note that in both cases a knot interior to a knot span of the local knot vector must be inserted p times to compute the extraction operator for element e. In Figure 4.8 we show the resulting extraction coefficients. In Appendix A.3, full extraction operators corresponding to elements 1, 10, 11 and 17 are presented.

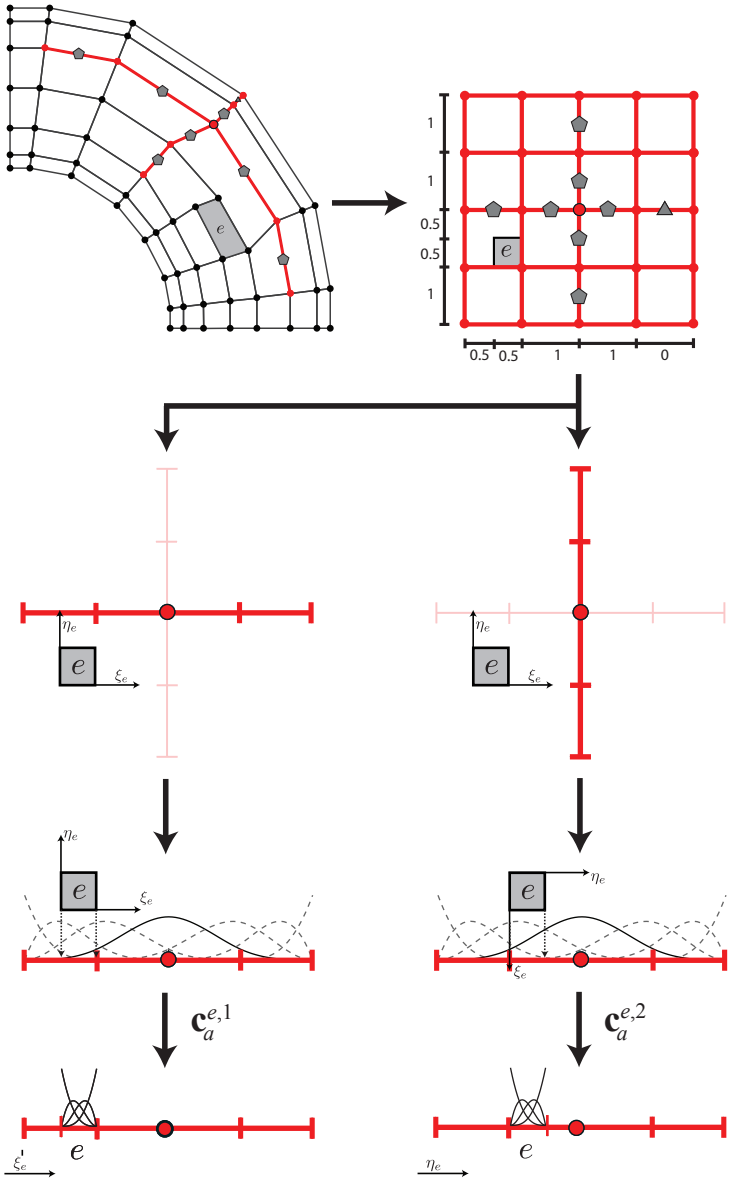


Figure 4.7: The extraction of a single T-spline basis function. Extraction is decoupled into two univariate extraction operations and then reassembled into a single row of $\mathbf{C}^e$. (The T-spline basis function is the one associated with control point 40, and element e corresponds to element 13 in the elemental T-mesh shown in Figure 3.9.)

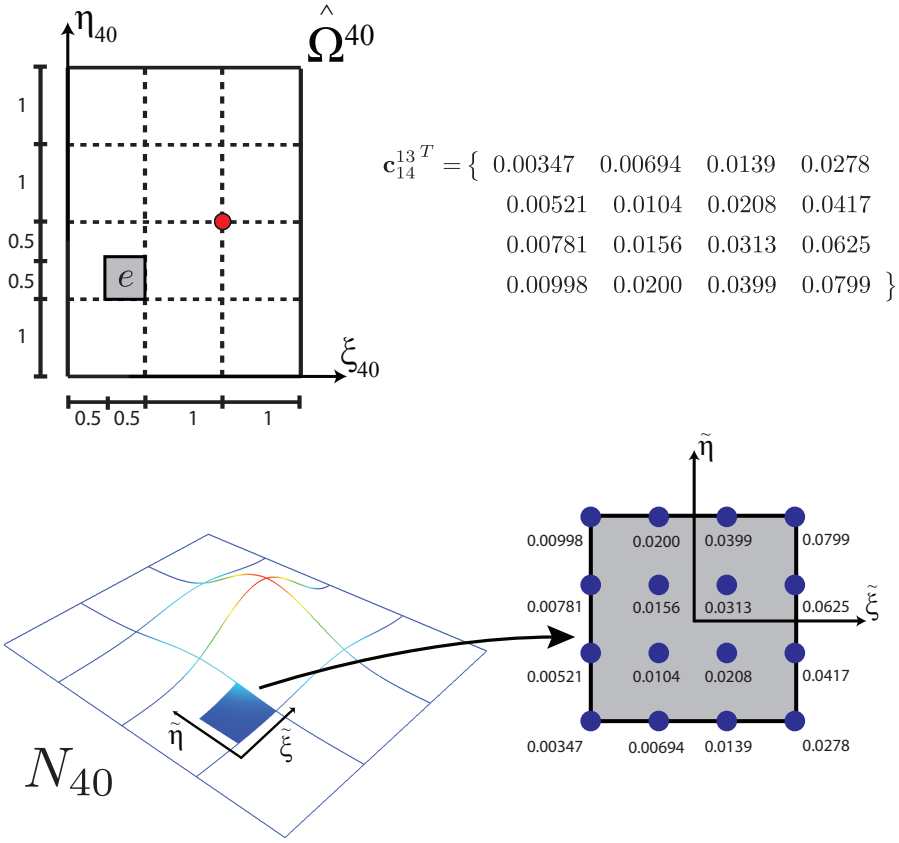


Figure 4.8: The result of extracting $N_{40}(\boldsymbol{\xi}_{40})$ over element e, where $e = 13$. On the top left, the position of element 13 in the local basis function space of $N_{40}(\boldsymbol{\xi}_{40})$ is shown. On the top right, Bézier extraction of N_{40} over element e (see Figure 4.7) generates $\mathbf{c}_{14}^{13\,T}$ where 14 is the local index of global control point 40 for element 13. In other words, $40 = \mathrm{IEN}(14, 13)$. Note that this represents a single row of the element extraction operator $\mathbf{C}^{13}$. On the bottom left, $N_{40}(\boldsymbol{\xi}_{40})$ is plotted. The shaded region is the restriction of $N_{40}(\boldsymbol{\xi}_{40})$ to the domain of element 13. On the bottom right, a close-up of element e is shown which portrays graphically the relationship between the extraction coefficients and Bernstein basis functions. Note that $N_{40}(\boldsymbol{\xi}_{40})|_{13} = \mathbf{c}_{14}^{13\,T}\,\mathbf{B}(\tilde{\boldsymbol{\xi}})$. See equation (4.13).

4.4 A Bézier extraction algorithm for T-splines

A Bézier extraction algorithm for T-splines consists of the following steps:

1. Infer the T-spline basis from the T-mesh. See Section 3.2.
2. Refine the T-mesh to construct the elemental T-mesh. See Section 3.4.2.
3. For a T-spline basis function determine the Bézier elements which are in its support. See Section 4.3.
4. For a T-spline basis function perform Bézier extraction. See Sections 4.3.1 and 4.3.2.
5. Repeat steps 3 and 4 for each T-spline basis function.

We note that the topological characterization of a T-mesh described in Section 3.1 is analogous to a quadrilateral mesh with hanging nodes. Thus, traditional quadrilateral meshing data structures and refinement schemes that allow hanging nodes are compatible with T-splines and can be used to perform the first three steps of the algorithm.

The fourth step is the most critical and is not mesh dependent. The primary operation in this step is the extraction of the univariate B-spline basis functions as described in Section 4.3.1. This routine is presented in Algorithm 1.

Algorithm 1 An algorithm to compute the univariate extraction operator rows corresponding to a single univariate B-spline basis function.

input Knot vector, $\Xi = \{\xi_1, \ldots, \xi_{p+2}\}$
Interior knots to be inserted into Ξ, $\mathtt{U} = \{\bar{\xi}_1, \ldots, \bar{\xi}_m\}$

Knot spans in Ξ where new knots will be inserted, spans $= \{\bar{s}_1, \ldots, \bar{s}_m\}$
Number of interior knots, m
Curve degree, p

output Element extraction operator rows c^e, $e = 1, 2, \ldots$,p+1+m

```
// Insert knots into the local knot vector and count the
// number of knots added to the front of the knot vector.
call compute_extended_knot_vector
    input: Ξ
    output: Ubar, nt

// Initialization variables:
a = p+1;
b = a+1;
nb = 1;
c^1 = 0;
c^1(nt+1) = 1;
mbar = p + 2 + nt + m;
ki = 1;
si = 1;
while  b < mbar  do
   c^(nb+1) = 0;// Initialize the next extraction operator row.

   // Count multiplicity of the knot at location b.
   add = 0;
   if  si <= m && spans(si) = ki  do
     mult = 0;
     add = 1;
     // Add the new knot to the knot vector.
     Ubar(b+1:mbar+p-m+si) = Ubar(b:mbar+p-m+si-1);
     Ubar(b) = U(si);
     si = si + 1;
   else
      ki = ki+1;
      i = b;
```

```
      while  b < m && Ubar(b+1) == Ubar(b)  do  b = b+1;
      mult = b-i+1;
    end

    if  mult < p  do
      numer = Ubar(b)-Ubar(a);
      for  j = p,p-1,...,mult+1  do
        alphas(j-mult) = numer / (Ubar(a+j+add)-Ubar(a));
      end
      r = p-mult;
      // Update the matrix coefficients for r new knots
      for  j=1,2,...,r  do
        save = r-j+1;
        s = mult+j;
        for  k=p+1,p,...,s+1  do
          alpha = alphas(k-s);
          c^nb(k) = alpha*c^nb(k) + (1.0-alpha)*c^nb(k-1);
        end
        if  b < m  do
          // Update overlapping coefficients of the
          // next operator row.
          c^nb+1(save) = c^nb(p+1);
        end
      end
      nb = nb + 1; // Finished with the current operator.
      if  b < m  do
        // Update indices for the next operator.
        a = b;
        b = b+1;
      end
    end
end
```

4.5 Incorporating $\mathbf{C}^e$ into the finite element formulation

Bézier extraction of T-splines generates a set of Bézier elements (written in terms of the Bernstein basis), the corresponding element extraction operators, $\mathbf{C}^e$, and the IEN array. This structure is identical to what was derived for NURBS in [23] and can be incorporated into a finite element formulation in an analogous fashion. Starting with an abstract weak formulation,

$$(W)\begin{cases} \text{Given } f\text{, find } u \in \mathcal{S} \text{ such that for all } w \in \mathcal{V} \\ \qquad a(w,u) = (w,f) \end{cases} \tag{4.14}$$

where $a(\cdot,\cdot)$ is a bilinear form and $(\cdot,\cdot)$ is the L^2 inner-product and $\mathcal{S}$ and $\mathcal{V}$ are the trial solution space and space of weighting functions, respectively, Galerkin's method consists of constructing finite-dimensional approximations of $\mathcal{S}$ and $\mathcal{V}$. In isogeometric analysis these finite-dimensional subspaces $\mathcal{S}^h \subset \mathcal{S}$ and $\mathcal{V}^h \subset \mathcal{V}$ are constructed from the T-spline basis which describes the geometry. The Galerkin formulation is then

$$(G)\begin{cases} \text{Given } f\text{, find } u^h \in \mathcal{S}^h \text{ such that for all } w^h \in \mathcal{V}^h \\ \qquad a(w^h,u^h) = (w^h,f) \end{cases} \tag{4.15}$$

In isogeometric analysis, the isoparametric concept is invoked, that is, the field in question is represented in terms of the geometric T-spline basis. We can write u^h and w^h as

$$w^h = \sum_{A=1}^{n} c_A R_A \tag{4.16}$$

$$u^h = \sum_{B=1}^{n} d_B R_B \tag{4.17}$$

where c_A and d_B are control variables. Substituting these into (4.15) yields the matrix form of the problem

$$\mathbf{K}\mathbf{d} = \mathbf{F} \tag{4.18}$$

where

$$\mathbf{K} = [K_{AB}], \tag{4.19}$$

$$\mathbf{F} = \{F_A\}, \tag{4.20}$$

$$\mathbf{d} = \{d_B\}, \tag{4.21}$$

$$K_{AB} = a\left(R_A, R_B\right), \tag{4.22}$$

$$F_A = \left(R_A, f\right). \tag{4.23}$$

The preceding formulation applies to scalar-valued partial differential equations, such as the heat conduction equation. The generalization to vector-valued partial differential equations, such as elasticity, follows standard procedures as described in [56].

4.5.1 The element shape function routine

As in standard finite elements, the global stiffness matrix, $\mathbf{K}$, and force vector, $\mathbf{F}$, can be constructed by performing integration over the Bézier elements to form element stiffness matrices and force vectors, $\mathbf{k}^e$ and $\mathbf{f}^e$, respectively, and assembling these into their global counterparts. The element form of (4.22) and (4.23) is

$$k^e_{ab} = a_e(R^e_a, R^e_b), \tag{4.24}$$

$$f^e_a = (R^e_a, f)_e \tag{4.25}$$

where $a_e(\cdot,\cdot)$ denotes the bilinear form restricted to the element, $(\cdot,\cdot)_e$ is the L^2 inner-product restricted to the element, and R_a^e are the element shape functions. The integration is usually performed by Gaussian quadrature. Since T-splines are not, in general, defined over a global parametric domain, all integrals are pulled back directly to the bi-unit parent element domain. No intermediate parametric domain is employed. This requires the evaluation of the global T-spline basis functions, their derivatives, and the Jacobian determinant of the pullback from the physical space to the parent element at each quadrature point in the parent element. These evaluations are done in an element shape function routine. Employing the element extraction operators we have that,

$$\mathbf{R}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{W}^e\mathbf{C}^e\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})}. \tag{4.26}$$

where

$$W^e(\tilde{\boldsymbol{\xi}}) = (\mathbf{w}^e)^T\mathbf{C}^e\mathbf{B}(\tilde{\boldsymbol{\xi}}). \tag{4.27}$$

The derivatives of $\mathbf{R}^e$ with respect to the local parent coordinates, $\tilde{\xi}^i$, are

$$\frac{\partial\mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial\tilde{\xi}^i} = \mathbf{W}^e\mathbf{C}^e\frac{\partial}{\partial\tilde{\xi}^i}\left(\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{W^e(\tilde{\boldsymbol{\xi}})}\right) = \mathbf{W}^e\mathbf{C}^e\left(\frac{1}{W^e(\tilde{\boldsymbol{\xi}})}\frac{\partial\mathbf{B}(\tilde{\boldsymbol{\xi}})}{\partial\tilde{\xi}^i} - \frac{\partial W^e(\tilde{\boldsymbol{\xi}})}{\partial\tilde{\xi}^i}\frac{\mathbf{B}(\tilde{\boldsymbol{\xi}})}{(W^e(\tilde{\boldsymbol{\xi}}))^2}\right). \tag{4.28}$$

To compute the derivatives with respect to the physical coordinates, (x_1^e, x_2^e, x_3^e), we apply the chain rule to get

$$\frac{\partial\mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial x_i^e} = \sum_{j=1}^{3}\frac{\partial\mathbf{R}^e(\tilde{\boldsymbol{\xi}})}{\partial\tilde{\xi}^j}\frac{\partial\tilde{\xi}^j}{\partial x_i^e}. \tag{4.29}$$

To compute $\partial\tilde{\boldsymbol{\xi}}/\partial\mathbf{x}^e$ we first compute $\partial\mathbf{x}^e/\partial\tilde{\boldsymbol{\xi}}$ using (3.13) and (4.28) and then take its inverse. Since we are integrating over the parent element we must also

compute the Jacobian determinant, J^e, of the mapping from the parent element to the physical space. It is computed as

$$J^e = \left| \frac{\partial \mathbf{x}^e}{\partial \tilde{\boldsymbol{\xi}}} \right|. \tag{4.30}$$

Higher-order derivatives can also be computed as described in Appendix B.

4.5.2 Finite element data structures

To employ an extracted T-spline in an existing finite element framework requires, in addition to the IEN array and extraction operators, the ID array and Bézier mesh. It is also useful to construct the LM array, which can be defined directly from the IEN and ID arrays.

Remark: The IEN, ID, and LM arrays are standard data processing arrays in finite element programs (see [56] for a full discussion). The acronyms mean, "element nodes," "destination," and "location matrix," respectively. The I's in IEN and ID are indicating the arrays are integer valued, corresponding to the implicit typing convention frequently used in Fortran programs. The implicit typing convention defines all variable names beginning with I, J, K, L, M, and N as integer valued, hence, LM is automatically integer valued. For the reader familiar with Fortran these remarks are unnecessary. However, Fortran is not common in CAGD, and many younger programmers are unfamiliar with it and its conventions. It is still widely utilized in computational mechanics, although the trend is toward C++. We note that traditionally Fortran uses a one-based indexing scheme while C++ uses a zero-based indexing scheme. Careful attention should be paid to this difference when implementing algorithms in each language.

4.5.2.1 Constructing the ID array

The ID array for the example domain Ω in Figure 3.1 is shown in Figure 4.9, where two degrees-of-freedom are assigned to every control point. This array maps the degree-of-freedom number (e.g., the direction index of a displacement component) and global control point number to the corresponding equation number in the global system. A zero value indicates a degree of freedom that is specified by the boundary conditions and for which the equation has been removed from the global system. For the domain Ω the horizontal and vertical displacements of control points 1, 9, 17, 29, 37, 44, and 51 are assumed to be specified.

4.5.2.2 Computing the Bézier mesh

Once the IEN array and element extraction operators have been computed, the control points for the Bézier elements can be computed by combining equations (3.9) and (4.2). In general, we obtain the Bézier control points $\mathbf{Q}^e$ and Bézier weights $\mathbf{w}^{b,e}$ for element e as

$$\mathbf{Q}^e = \left(\mathbf{W}^{b,e}\right)^{-1} \left(\mathbf{C}^e\right)^T \mathbf{W}^e \mathbf{P}^e, \tag{4.31}$$

$$\mathbf{w}^{b,e} = \left(\mathbf{C}^e\right)^T \mathbf{w}^e, \tag{4.32}$$

where $\mathbf{P}^e$ are the T-spline control points, $\mathbf{W}^{b,e}$ is the diagonal matrix of Bézier weights, and $\mathbf{W}^e$ is the diagonal matrix of T-spline weights, $\mathbf{w}^e$. Figure 4.10 shows the result of (4.31) for Bézier elements 1, 10, 11, 17 (see Appendix A.2 for the element control point coordinates). The T-spline control points which contribute to the location of the Bézier element control points are indicated by the ◯'s. Each T-spline element (see Section 3.4.5) has an equivalent representation as an extracted Bézier element. In other words

$$\mathbf{x}^e(\tilde{\boldsymbol{\xi}}) = \mathbf{x}^{b,e}(\tilde{\boldsymbol{\xi}}) = \frac{1}{W^{b,e}(\tilde{\boldsymbol{\xi}})} (\mathbf{Q}^e)^T \mathbf{W}^{b,e} \mathbf{B}(\tilde{\boldsymbol{\xi}}). \tag{4.33}$$

where $W^{b,e}(\tilde{\boldsymbol{\xi}}) = \mathbf{w}^{b,e\,T}\,\mathbf{B}(\tilde{\boldsymbol{\xi}})$. We have defined three "meshes" for the T-spline example in Figure 3.1 – the elemental T-mesh, the Bézier control mesh, and the Bézier physical mesh. Figure 4.11 shows a representative element in each of these meshes and how they are related using $\mathbf{C}^e$. It is important to remember that the Bézier physical mesh defines the set of finite elements used in computations. The extraction operator corresponding to each of these elements is then used to constrain the Bézier degrees-of-freedom such that the original smoothness of the T-spline is present in the finite element solution.

We have implemented a finite element code which incorporates Bézier elements and the extraction operator and have tested this approach extensively. In all cases where a comparison exists, it gives identical results to those obtained using isogeometric implementations which do not utilize T-spline extraction. We have also found that this approach greatly reduces finite element assembly times and reduces the complexity of parallelizing isogeometric codes. Another important practical advantage to this approach is that it allows an analyst to compute with T-splines without understanding the details of T-spline technology. The extraction paradigm provides a simple interface which is readily accessible to most finite element practitioners.

ID array:

		Global control point number (A)														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Degree-of-freedom	1	0	1	3	5	7	9	11	13	0	15	17	19	21	23	25
number (i)	2	0	2	4	6	8	10	12	14	0	16	18	20	22	24	26

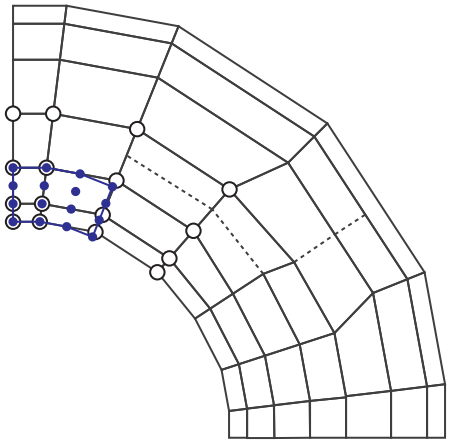
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
27	0	29	31	33	35	37	39	41	43	45	47	49	0	51
28	0	30	32	34	36	38	40	42	44	46	48	50	0	52

31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
53	55	57	59	61	63	0	65	67	69	71	73	75	0	77
54	56	58	60	62	64	0	66	68	70	72	74	76	0	78

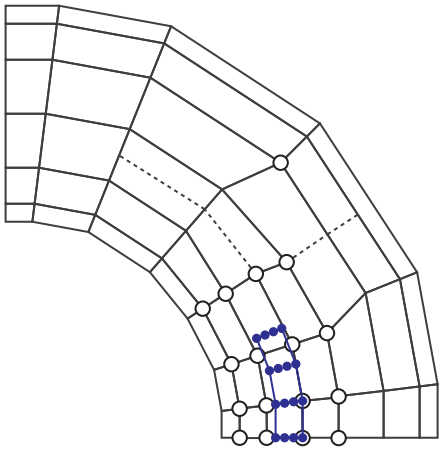
46	47	48	49	50	51	52	53	54	55	56	57
79	81	83	85	87	0	89	91	93	95	97	99
80	82	84	86	88	0	90	92	94	96	98	100

$$P = \mathrm{ID}(i, A)$$

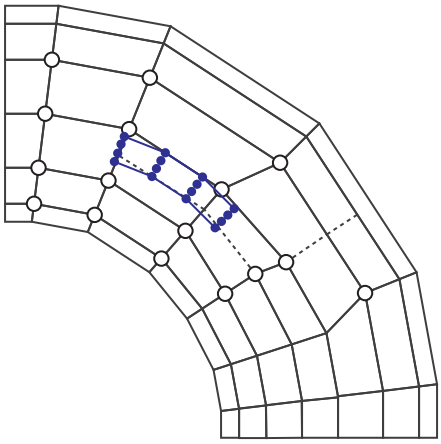
Figure 4.9: The ID array maps the degree-of-freedom number (i.e., direction index of the displacement component) and global control point number to the corresponding equation number in the global system. For this example the horizontal and vertical displacements of control points 1, 9, 17, 29, 37, 44, and 51 are specified by boundary conditions. The global equations corresponding to these degrees-of-freedom are removed from the system through the ID array. The LM array is computed as follows: $P = \mathrm{LM}(i, a, e) = \mathrm{ID}(i, \mathrm{IEN}(a, e))$.



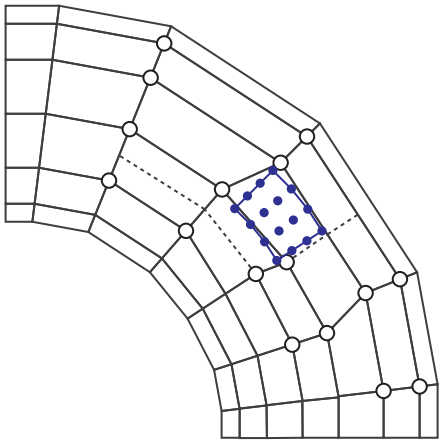
Bézier control element 1



Bézier control element 10



Bézier control element 11



Bézier control element 17

Figure 4.10: The extraction operators and IEN array can be used to construct the Bézier control elements. For each control element e the ◯'s indicate the global T-spline control points which influence the location of Bézier control points, indicated by the •'s. The global control points that influence each control element are determined by the IEN array, and the location of the element control points is computed with the element extraction operator $\mathbf{C}^e$.

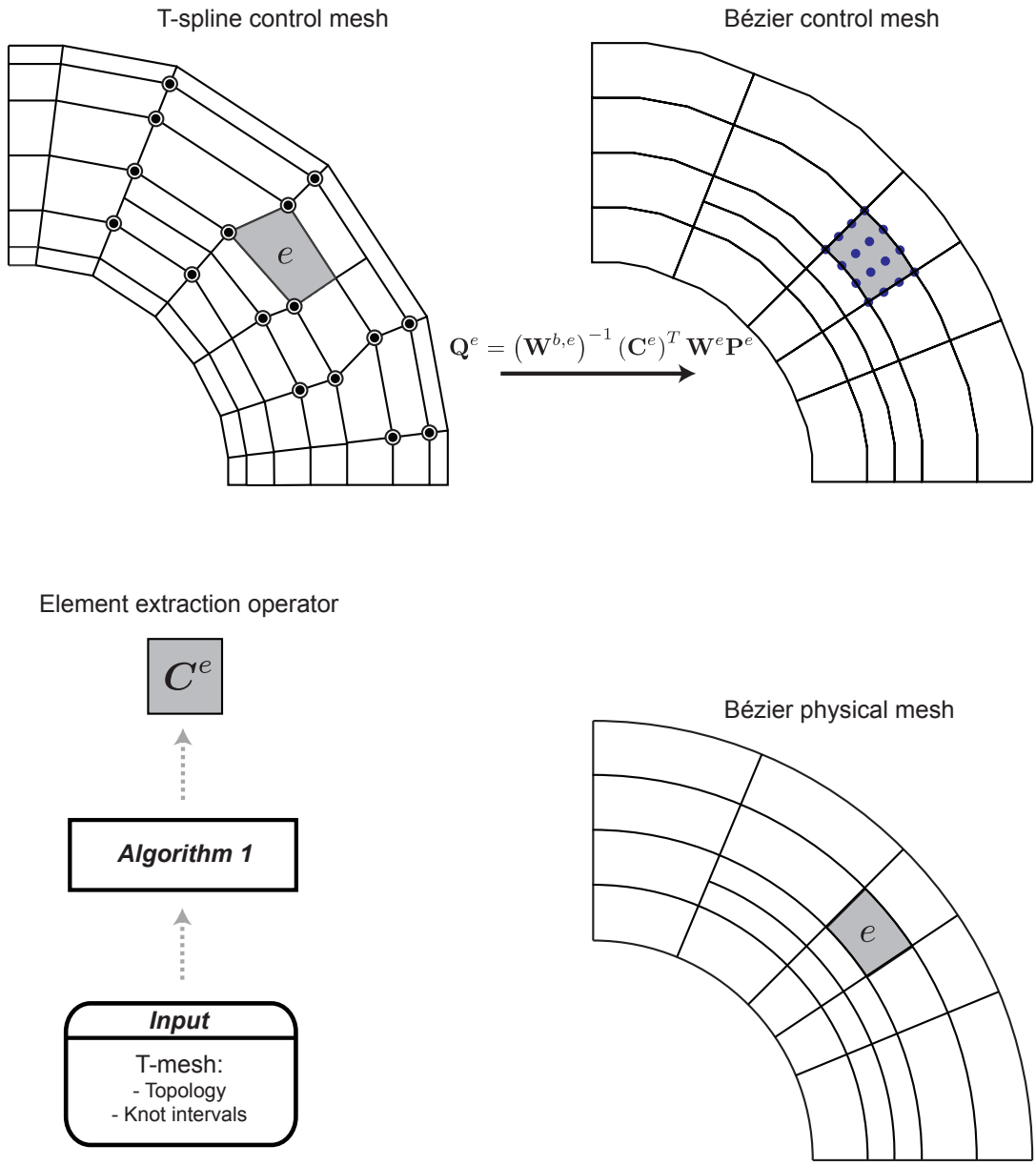


Figure 4.11: Various "meshes" of the T-spline example.

Chapter 5

Analysis-suitable T-splines and local refinement

Analysis-suitable T-splines form a practically useful subset of T-splines. Analysis-suitable T-splines maintain the important mathematical properties of the NURBS basis while providing an efficient and highly localized refinement capability. All T-splines possess the following properties:

- The basis constitutes a partition of unity [70] (see Section 5.5.)
- Each basis function is non-negative.
- An affine transformation of an analysis-suitable T-spline is obtained by applying the transformation to the control points. We refer to this as affine covariance. This implies that all "patch tests" (see [56]) are satisfied *a priori*.
- They obey the convex hull property.
- They can be locally refined.

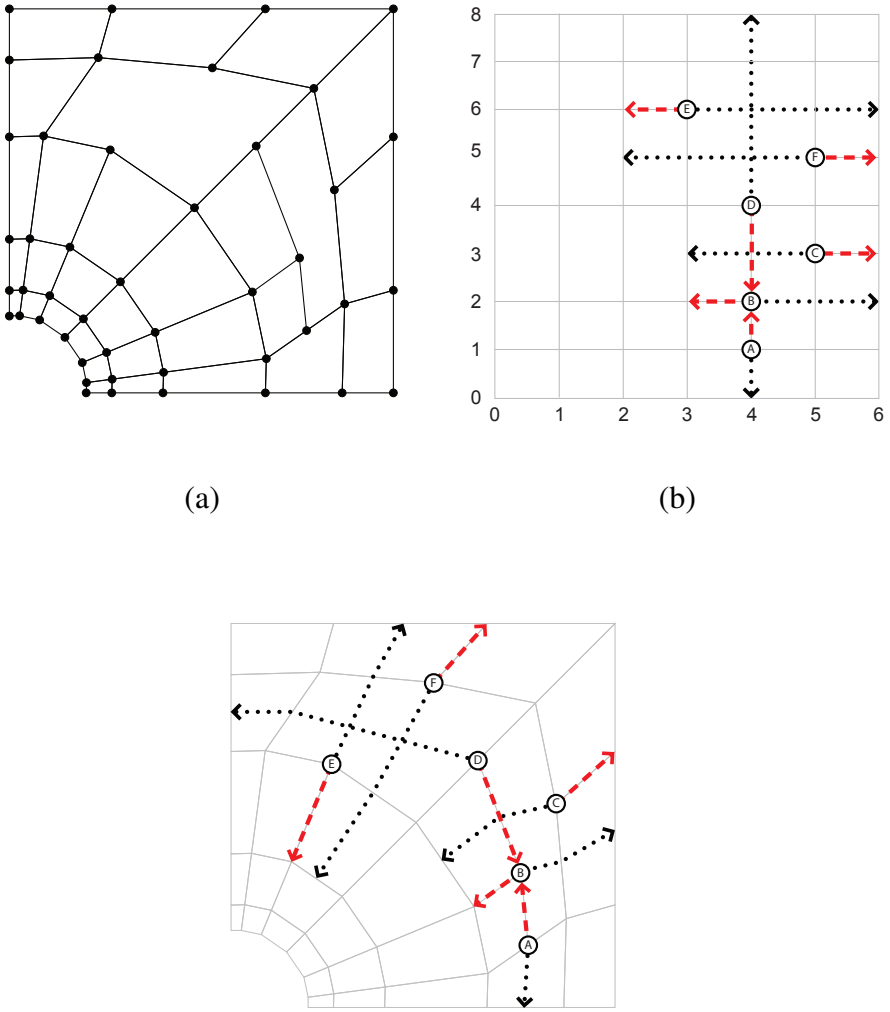
While most (but not all [27]) T-splines are also linearly independent, analysis-suitable T-splines are *always* linearly independent, for any choice of knot intervals [71]. Analysis-suitable T-spline spaces are defined over a mildly restricted set of allowable T-mesh topologies. This topological restriction can be described elegantly in terms of *T-junction extensions*.

5.1 T-junction extensions

A T-junction extension is normally composed of a face and edge extension (if one exists.) We define face and edge extensions to be closed, directed line segments which originate at a T-junction. An extended T-mesh T_{ext} is formed by adding all T-junction extensions to a T-mesh T. The extended T-mesh, T_{ext}, for the T-mesh in Figure 5.1a is shown in Figures 5.1b and 5.1c, respectively. The dotted black arrows are face extensions and the dashed red arrows are edge extensions. The T-junctions are denoted by large circles.

Figure 5.1b shows the T-junction extensions in the *index space* [9] of the T-mesh as described in Section 2.5. In the index space, the direction of traversal and orientation of a T-junction extension can be uniquely established. Figure 5.1c shows the set of T-junction extensions drawn on the T-mesh in physical space. We often drop the distinction between the index and physical space representation for T-junction extensions and use the physical space representation exclusively.

A T-junction extension is formed in a manner similar to what was described for the construction of local knot interval vectors in Section 3.2.1. A face extension is created by marching from the T-junction, in the direction of a missing edge, until two perpendicular edges or vertices are intersected. The direction of each extension is always away from its T-junction. An edge extension is then formed *only* if an edge is attached to the T-junction in the opposite direction. If so, the extension is formed by marching to the edge's opposite vertex. Since T-junction extensions are closed line segments, a horizontal and vertical extension can intersect either on the interior of both extensions or at the endpoint of one extension or both extensions.



(a) (b)

(c)

Figure 5.1: The extended T-mesh, T_{ext}, in the index and physical space. The dotted arrows (in black) represent face extensions and the dashed arrows (in red) represent edge extensions. The T-junctions are denoted by large circles. (a) A T-mesh T with six T-junctions. (b) The extended T-mesh formed from T and the T-junction extensions in index space. The indexing of the knot structure of T is along the bottom and left. (c) The extended T-mesh in physical space. Note that the directionality of each extension is always determined in the index space of the T-mesh.

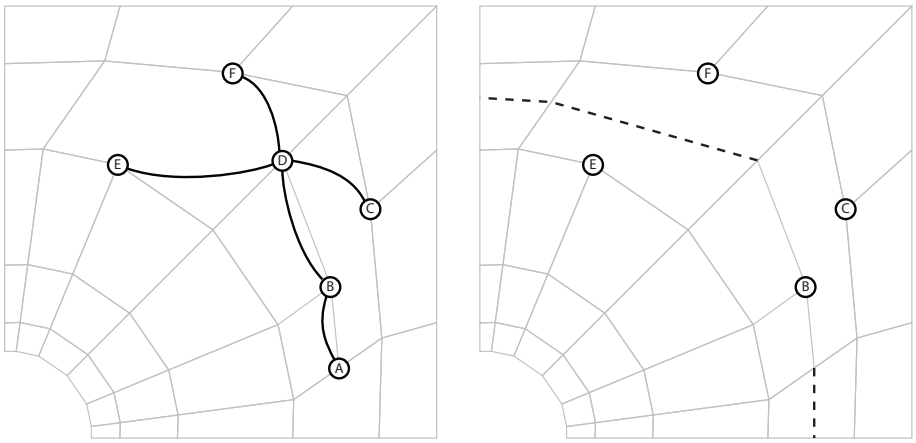
As an example, consider T-junction E in Figure 5.1b. The face extension points to the right and intersects the two vertical edges corresponding to indices 5 and 6 along the bottom. Since T-junction E is connected to an edge in the direction opposite the face extension we also form an edge extension along that edge. The edge extension points to the left and intersects the vertical edge corresponding to index 2 along the bottom. The T-junction extension for T-junction E is composed of the face *and* edge extension.

5.2 The extension graph

Intersecting T-junction extensions in an extended T-mesh T_{ext} can be visualized using an undirected graph. We call this graph the extension graph and denote it by $E(\mathsf{T}_{ext})$. Each node in E corresponds to a single T-junction extension in T_{ext}. If two extensions in T_{ext} intersect then an edge is drawn between the corresponding nodes in E. The extension graph for the extended T-mesh in Figure 5.1b is shown in Figure 5.2a. In this case there are five intersections represented by the five edges in the graph.

5.3 Analysis-suitable definition

An analysis-suitable T-spline is one whose extended T-mesh is analysis-suitable. An analysis-suitable extended T-mesh is one where no T-junction extensions intersect. In other words, $E(\mathsf{T}_{ext})$ is an empty graph (no edges in the graph). We denote an analysis-suitable T-spline space by $\mathfrak{T}_s$ and analysis-suitable T-mesh by T_s. The T-mesh in Figure 5.1a is not analysis-suitable. This can be seen by inspecting the extension graph in Figure 5.2a which has five edges. By adding the dashed edges in Figure 5.2b the T-mesh becomes analysis-suitable because its



(a) (b)

Figure 5.2: The extension graph, $\mathrm{E}(\mathsf{T}_{ext})$. (a) The extension graph $E(\mathsf{T}_{ext})$ corresponding to T_{ext} in Figures 5.1b and 5.1c. The five edges in the graph correspond to the five intersections between T-junction extensions in T_{ext}. (b) The T-mesh in Figure 5.1a can be made analysis-suitable by adding the bold dashed edges. The extension graph for this new T-mesh is empty (no edges.)

extension graph is empty.

5.4 The analysis-suitable elemental T-mesh

For analysis-suitable T-splines, the elemental T-mesh, T_{elem} (see Section 3.4.5), can be formed by simply adding the face extensions to T [70]. This greatly simplifies the construction of T_{elem} since it is *not* necessary to inspect the knot lines in the T-spline basis.

5.5 Partition of unity

An analysis-suitable T-spline basis forms a partition of unity. In other words, $\sum_{A=1}^{n} N_A = 1$. The partition of unity property is important for both geometry and analysis because it assures affine covariance and exact satisfaction of all patch tests. In the context of T-splines, the partition of unity property can be described in two ways. The equation for *any* T-spline surface is

$$T = \frac{\sum_{A=1}^{n} \mathbf{P}_A w_A N_A}{\sum_{A=1}^{n} w_A N_A} \tag{5.1}$$

where the $\mathbf{P}_A$ are control points, w_A are weights, and N_A are blending functions. This equation can also be written as

$$T = \sum_{A=1}^{n} \frac{w_A N_A}{\sum_{B=1}^{n} w_B N_B} \mathbf{P}_A = \sum_{A=1}^{n} R_A \mathbf{P}_A \tag{5.2}$$

in which case the R_A may be *rational* T-spline blending functions. It is clear that the R_A always sum to one, regardless of the choice of w_A. In general, affine covariance only requires that the R_A form a partition of unity, not the N_A. However, it is shown in [70] that for an analysis-suitable T-spline with all $w_A = 1$, $R_A \equiv N_A$. Thus, when we say that analysis-suitable T-splines form a partition of unity, we mean that both the N_A and R_A sum to one. This stronger notion of partition of unity is also a property of NURBS.

5.6 Local refinement of analysis-suitable T-splines

The T-spline local refinement algorithm presented in [104] may add many superfluous control points to a T-mesh during refinement. Additionally, using this algorithm to locally refine an analysis-suitable T-spline often results in a refined T-spline which is *not* analysis-suitable.

This behavior can be attributed to the generality of the algorithm, which is designed to operate on *any* T-spline. No assumptions are made about the topological characteristics of the underlying T-mesh or space. By restricting ourselves to analysis-suitable T-splines, however, we can leverage the structure of the T-mesh to develop a simple local refinement algorithm which only introduces a minimal number of superfluous control points and preserves the properties of an analysis-suitable space.

5.6.1 Local refinement fundamentals

The set of all T-splines with the same T-mesh topology, T, and knot interval configuration is called a *T-spline space* [104]. We denote a T-spline space by $\mathcal{T}$, where the number of T-spline control points in T is n. While the notation $\mathcal{T}^1 \subseteq \mathcal{T}^2$ will be used in the conventional set-theoretic sense, the notation $\mathsf{T}^1 \subseteq \mathsf{T}^2$ will indicate that T^2 can be created by adding vertices and edges to T^1, and appropriately modifying the knot intervals on any edges which are split. In the context of finite element analysis, vertices and edges are usually added by subdividing T-mesh elements.

If $\mathcal{T}^1 \subseteq \mathcal{T}^2$, $\mathcal{T}^1$ and $\mathcal{T}^2$ are said to be *nested* and $\mathcal{T}^2$ is a *local refinement* of $\mathcal{T}^1$. In Figure 5.3, a T-mesh, T^1, (solid circles and lines) and corresponding T-spline space, $\mathcal{T}^1$, are locally refined through the addition of control points and edges (hollow circles and dashed edges). In this case, $\mathsf{T}^1 \subseteq \mathsf{T}^2 \rightarrow \mathcal{T}^1 \subseteq \mathcal{T}^2$.

5.6.1.1 Basis function refinement

In basis function refinement [90, 104], knots are added to the local knot vector of a cubic B-spline basis function, $N(\xi \,|\, \Xi)$, where $\Xi = \{\xi_1, \xi_2, \ldots, \xi_5\}$, to form a knot vector, $\bar{\Xi}$, of length m. N can then be written as a linear combination

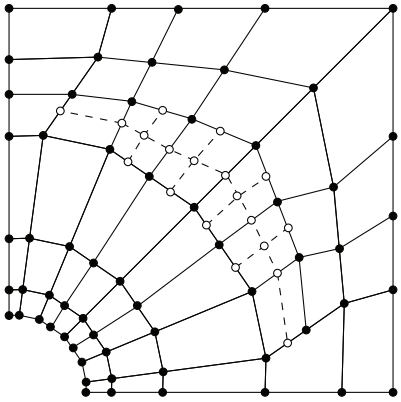


Figure 5.3: A T-mesh, T^1, (solid circles and lines) and T-spline space, $\mathcal{T}^1$, is locally refined through the addition of control points and edges (hollow circles and dashed edges). In this case, $\mathsf{T}^1 \subseteq \mathsf{T}^2 \rightarrow \mathcal{T}^1 \subseteq \mathcal{T}^2$.

of the $m-4$ B-spline basis functions defined over substrings of length 5 in $\bar{\Xi}$.

For the case $m = 6$, $N(\xi \,|\, \Xi)$ is split by inserting a single knot $\bar{\xi}$ into Ξ where $\xi_i \leq \bar{\xi} \leq \xi_{i+1}$. This splits the basis function into two scaled basis functions:

$$N(\xi \,|\, \xi_1, \ldots, \xi_5) = aN(\xi \,|\, \xi_1, \ldots, \xi_i, \bar{\xi}, \xi_{i+1}, \ldots, \xi_4) + bN(\xi \,|\, \xi_2, \ldots, \xi_i, \bar{\xi}, \xi_{i+1}, \ldots, \xi_5) \tag{5.3}$$

where

$$a = \begin{cases} \frac{\bar{\xi}-\xi_1}{\xi_5-\xi_1} & \text{for } k < \xi_4 \\ 1 & \text{for } k \geq \xi_4 \end{cases} \tag{5.4}$$

and

$$b = \begin{cases} \frac{\xi_5-\bar{\xi}}{\xi_5-\xi_1} & \text{for } k > \xi_2 \\ 1 & \text{for } k \leq \xi_2 \end{cases}. \tag{5.5}$$

The refinement equations for the case $m > 6$ can be derived through repeated application of these equations.

A T-spline basis function, $N(\boldsymbol{\xi} \,|\, \boldsymbol{\Xi})$, can undergo knot insertion in either parametric direction by inserting a knot into the corresponding local knot vector

and then applying the refinement equations. This results in two scaled T-spline basis functions which sum to the original. Further knot insertion into these resultant scaled basis functions yields a set of scaled basis functions which also sum to the original.

5.6.1.2 The refinement operator $\mathbf{M}$

If $\mathcal{T}^1 \subseteq \mathcal{T}^2$, each T-spline basis function, $N_A^1 \in \mathcal{T}^1$, can be expressed uniquely as a linear combination of the T-spline basis functions, $N_B^2 \in \mathcal{T}^2$, as

$$N_A^1 = \sum_{B=1}^{n_2} m_{A,B} N_B^2 \tag{5.6}$$

where n_2 is the number of control points in $\mathcal{T}^2$ and the $m_{A,B}$ are determined by knot insertion as described in Section 5.6.1.1. This relationship can be written in matrix-vector notation as

$$\mathbf{N}^1 = \mathbf{M}\mathbf{N}^2 \tag{5.7}$$

where $\mathbf{N}^1 = \{N_1^1, N_2^1, \ldots, N_{n_1}^1\}^T$ is the column vector of T-spline basis functions, $N_A^1 \in \mathcal{T}^1$, $\mathbf{N}^2 = \{N_1^2, N_2^2, \ldots, N_{n_2}^2\}^T$ is the column vector of T-spline basis functions, $N_B^2 \in \mathcal{T}^2$, and $\mathbf{M}$ is an $n_1 \times n_2$ matrix with elements $m_{A,B}$. We call $\mathbf{M}$ the refinement operator.

5.6.2 Analysis-suitable nesting theory

We say T_{ext}^2 is a refinement of T_{ext}^1 (denoted $\mathsf{T}_{ext}^1 \hat{\subseteq} \mathsf{T}_{ext}^2$) if $\mathsf{T}^1 \subseteq \mathsf{T}^2$ and no face extension endpoint in T_{ext}^2 corresponds to a point in the interior of a face extension in T_{ext}^1 of the same directionality. Then, if $\mathcal{T}_s^1$ and $\mathcal{T}_s^2$ are analysis-suitable T-spline spaces, $\mathsf{T}_{ext}^1 \hat{\subseteq} \mathsf{T}_{ext}^2 \rightarrow \mathcal{T}_s^1 \subseteq \mathcal{T}_s^2$. We note that a rigorous proof of this result is nearly complete [69]. Note that we compare face extensions in the index space of T_{ext}^2.

Nestedness between two T-spline spaces, $\mathfrak{T}_s^1$ and $\mathfrak{T}^2$ (not necessarily analysis-suitable), can be visualized using a simple modification of the extension graph described in Section 5.2. We call this graph the *coupled extension graph*, $E(\mathsf{T}_{ext}^{1\to 2})$. $\mathsf{T}_{ext}^{1\to 2}$, called a *coupled extended T-mesh*, is constructed by adding the face extensions of T_{ext}^1 to T_{ext}^2.

To construct $E(\mathsf{T}_{ext}^{1\to 2})$ we augment $E(\mathsf{T}_{ext}^2)$ by adding an additional edge to the graph if a T-junction face extension endpoint in T_{ext}^2 is in the interior of a face extension from T_{ext}^1 of the same directionality. In that case, a "loop edge" is drawn from the corresponding node in $E(\mathsf{T}_{ext}^2)$ to itself creating a small loop in the graph. If T_{ext}^2 is analysis-suitable ($E(\mathsf{T}_{ext}^2)$ is empty) then the only edges which exist in $E(\mathsf{T}_{ext}^{1\to 2})$ are "loop edges." The non-existence of loop edges is a necessary condition that $\mathsf{T}_{ext}^1 \hat{\subseteq} \mathsf{T}_{ext}^2$. If $E(\mathsf{T}_{ext}^{1\to 2})$ is an empty graph then $\mathfrak{T}_s^1 \subseteq \mathfrak{T}^2$. The *weight* of a coupled extension graph is the number of edges in the graph and is denoted by $W(E)$. The weight of a node is the number of edges touching the node.

Figure 5.4 illustrates the construction of a coupled extension graph and its use in determining nestedness. An analysis-suitable extended T-mesh T_{ext}^1 is shown in Figure 5.4a and T_{ext}^2 is shown Figure 5.4b. We construct the coupled extended T-mesh $\mathsf{T}_{ext}^{1\to 2}$ in Figure 5.4c. Notice that in this case the only face extension from T_{ext}^1 which is visible (a light gray dotted arrow in a box) corresponds to T-junction extension B in Figure 5.4a. This indicates that the endpoint of T-junction extension C in T_{ext}^2 (see Figure 5.4b) is in the interior of T-junction extension B in T_{ext}^1 (see Figure 5.4a.) Figure 5.4d shows $E(\mathsf{T}_{ext}^{1\to 2})$. Since the graph is not empty, $\mathfrak{T}_s^1 \not\subseteq \mathfrak{T}^2$. In fact, the edges between different nodes means the underlying extension graph $E(\mathsf{T}_{ext}^2)$ is not empty which implies that $\mathfrak{T}^2$ is not analysis-suitable. The loop edge which begins and ends at node C indicates that $\mathsf{T}_{ext}^1 \hat{\not\subseteq} \mathsf{T}_{ext}^2$.

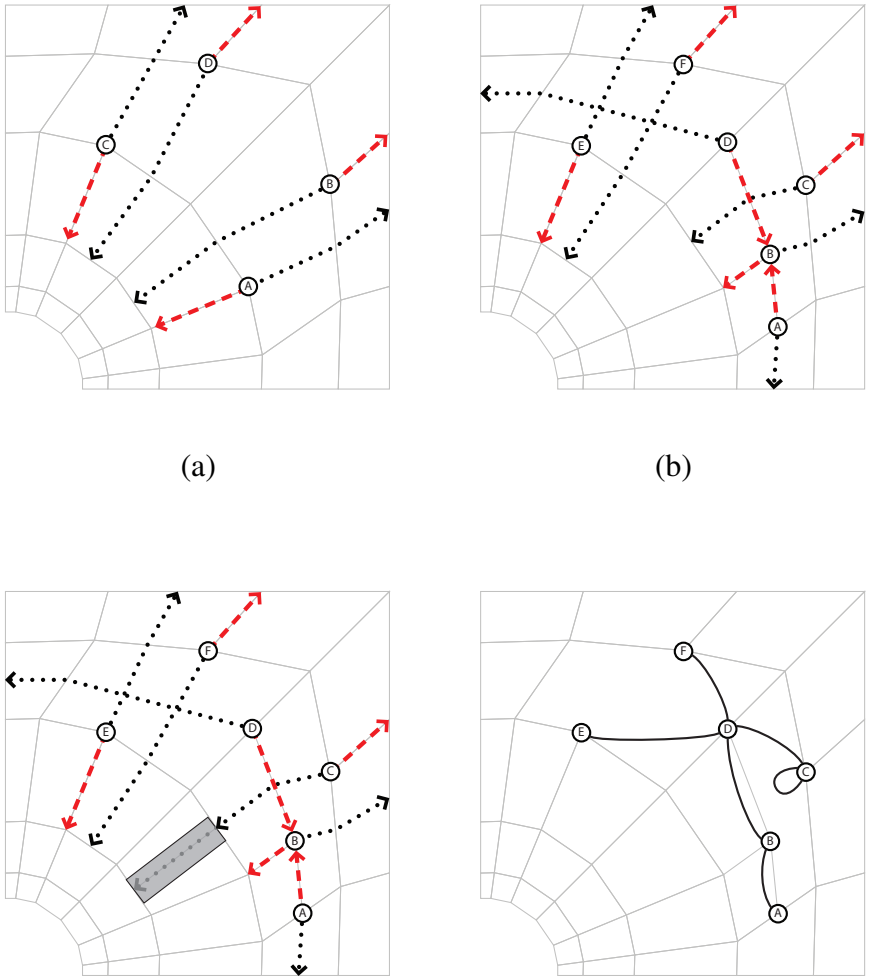


Figure 5.4: Determining nestedness. (a) An analysis-suitable extended T-mesh T^1_{ext}. (b) An extended T-mesh T^2_{ext} (not analysis-suitable). (c) Superimposing the face extensions from (a) on T^2_{ext} to form $\mathsf{T}^{1\to 2}_{ext}$. Notice that the only face extension from T^1_{ext} which is visible is in the light gray box. (d) The coupled extension graph for (c). The graph contains edges which implies that $\mathcal{T}^1_s \not\subseteq \mathcal{T}^2$.

5.6.3 A local refinement algorithm

The example in Figure 5.4 motivates a simple approach to local refinement. First, create $\mathsf{T}^2 \supseteq \mathsf{T}^1_s$. As in standard finite element analysis, this is usually done by subdividing a set of T-mesh elements in T^1_s. The T-mesh elements are often selected so as to reduce error in the finite element solution. If $E(\mathsf{T}^{1\to 2}_{ext})$ is not empty, we must add some additional control points and edges to T^2 to cause $E(\mathsf{T}^{1\to 2}_{ext})$ to be empty.

To illustrate, Figure 5.5a shows the T-mesh T^2 from Figure 5.4b, where $\mathcal{T}^1_s \not\subseteq \mathcal{T}^2$ eventhough $\mathsf{T}^1_s \subseteq \mathsf{T}^2$. This can be seen by inspecting the coupled extension graph in Figure 5.4d. To ensure nestedness, additional refinement of T^2 must be performed. Figures 5.5b through 5.5d show three possible analysis-suitable refinements of T^2 where the resulting coupled extension graph $E(\mathsf{T}^{1\to 2}_{ext})$ is empty. The dashed edges and open circles in Figure 5.5 are T-mesh edges and vertices, respectively, added during local refinement. The minimum of the three refinements is shown in Figure 5.5d, where only 6 vertices and 8 edges have been added. We note that Figure 5.5b is a NURBS refinement.

This example raises the question of how to devise an algorithm for automatically finding the fewest additional control points and edges that will cause $E(\mathsf{T}^{1\to 2}_{ext})$ to be empty. Finding the minimal number is an NP-hard problem, so for efficiency, our algorithm uses the following greedy strategy that only provides an approximate minimum. The following steps constitute our analysis-suitable local refinement algorithm:

1. Create $\mathsf{T}^2 \supseteq \mathsf{T}^1_s$.

2. Using T^1_s and T^2, form $E(\mathsf{T}^{1\to 2}_{ext})$.

3. Of all possible insertion edges, add one into T^2 for which the weight of the resulting $E(\mathsf{T}^{1\to 2}_{ext})$ is smallest. An *insertion edge* has a vertex that is a T-junction in T^2 and the corresponding node in the coupled extension graph has non-zero weight (see Figure 5.6.)

4. Repeat Step 3 until the weight of $E(\mathsf{T}^{1\to 2}_{ext})$ is zero.

5. Compute the refinement operator $\mathbf{M}$, if desired. See Section 5.7.

Only one insertion edge may be inserted at a time during each iteration of the refinement algorithm. Also, there are cases for which the weight will stay the same or increase after the addition of the optimal insertion edge. It should be noted that the algorithm will always terminate, because in the limit, if all T-junctions are extended all the way to a boundary edge, a NURBS is created.

We first demonstrate Steps 1 through 4 of analysis-suitable local refinement on a simple example. Step 5 is explained in detail in Section 5.7. We begin with the analysis-suitable T-mesh T^1_s shown in Figure 5.7a. In Figure 5.7b, T^1_s is refined by subdividing several T-mesh elements. The coupled extension graph $E(\mathsf{T}^{1\to 2}_{ext})$ can then be constructed from the coupled extended T-mesh $\mathsf{T}^{1\to 2}_{ext}$ as shown in Figures 5.7d and 5.7c, respectively. Notice that the weight of the graph is 4 so nesting is not assured between $\mathcal{T}^1_s$ and $\mathcal{T}^2$.

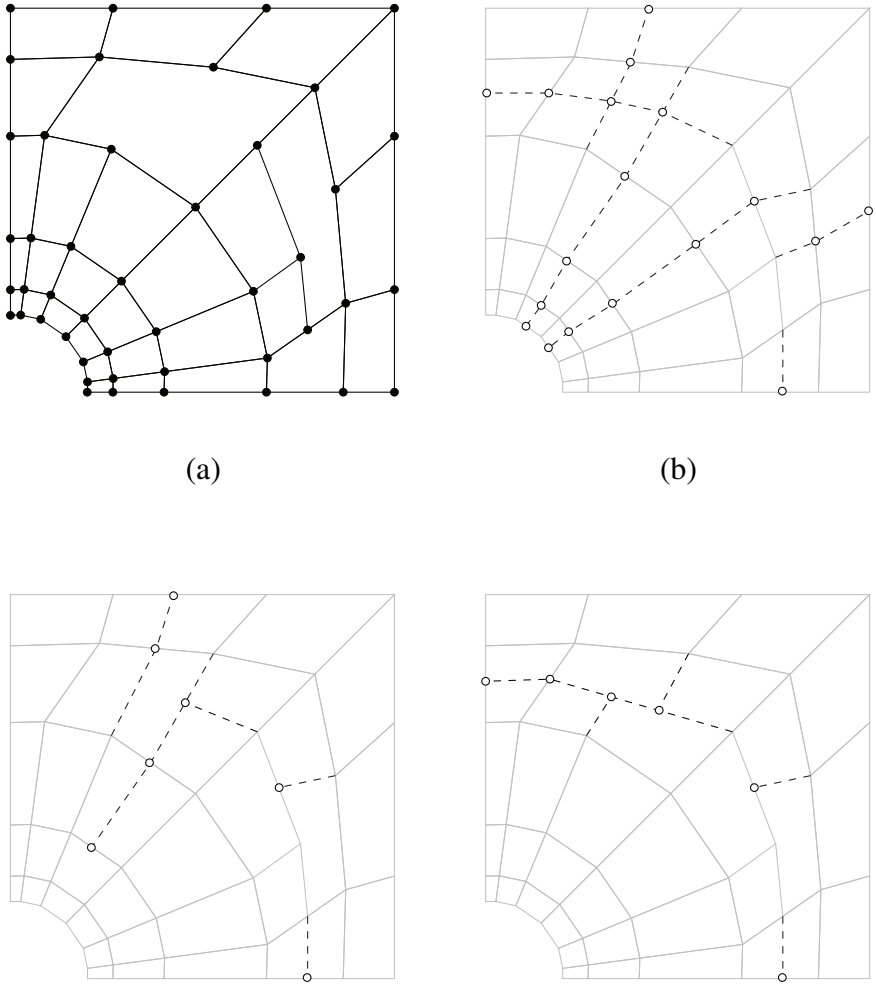
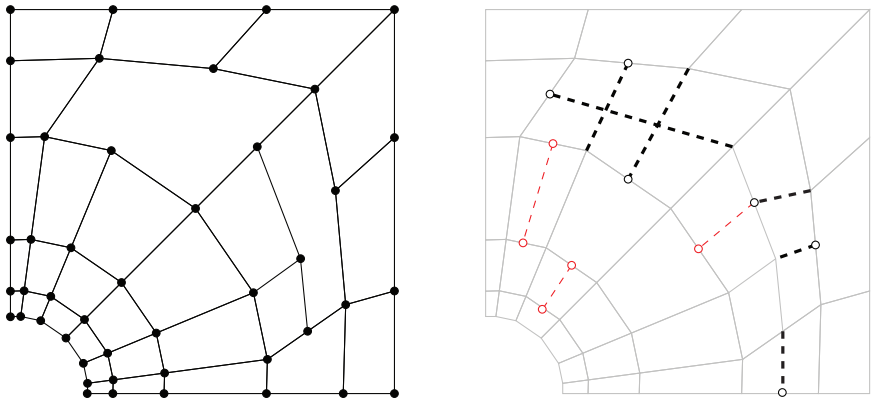


Figure 5.5: Several analysis-suitable local refinements for the example in Figure 5.4. The dashed edges and open circles are T-mesh edges and vertices, respectively, added during local refinement. (a) The T-mesh T^2 from Figure 5.4b. (b) 18 vertices and 19 edges have been added. (c) 7 vertices and 8 edges have been added. (d) 6 vertices and 8 edges have been added.



(a) (b)

Figure 5.6: T-mesh edges which may be inserted during Step 3 of the local refinement algorithm. (a) The T-mesh T^2. (b) The thick black dashed lines are edges which can be inserted into T^2. Notice that each of these edges has a vertex which is a T-junction in T^2 and the corresponding node in the coupled extension graph has non-zero weight (see Figure 5.4d.) The thick black dashed lines are called insertion edges. The thin red dashed lines are examples of T-mesh edges which cannot be inserted to form a refined T-mesh.

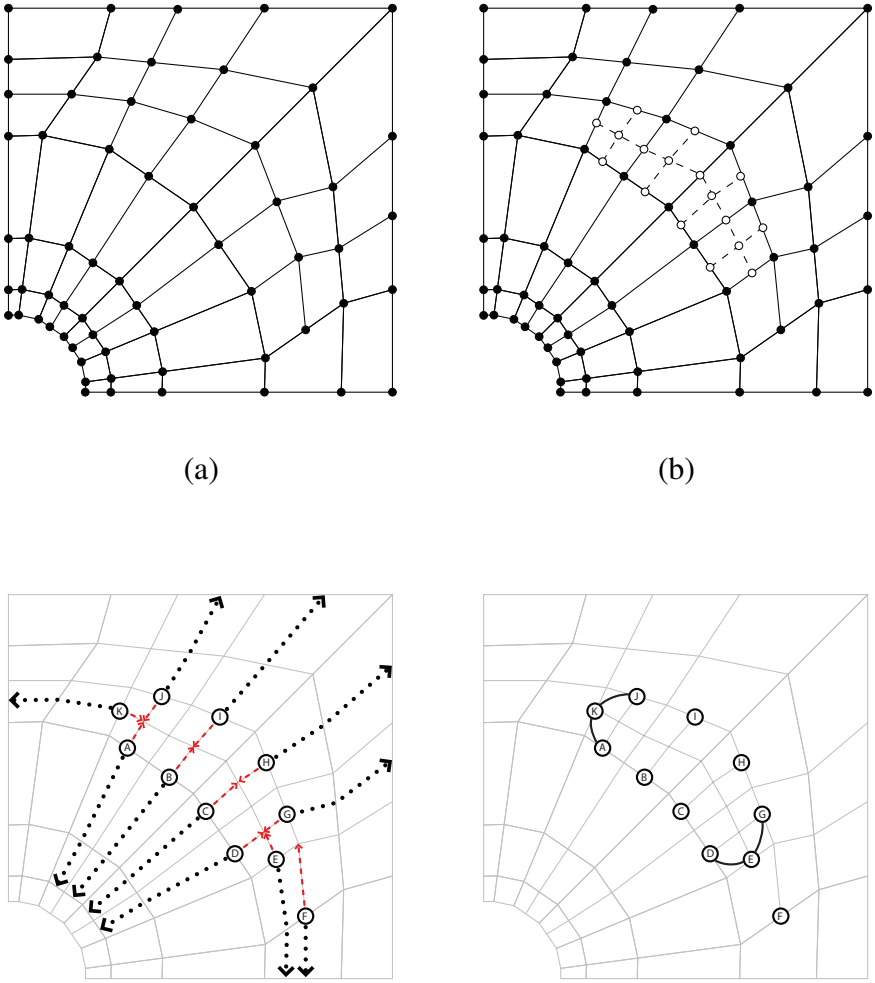
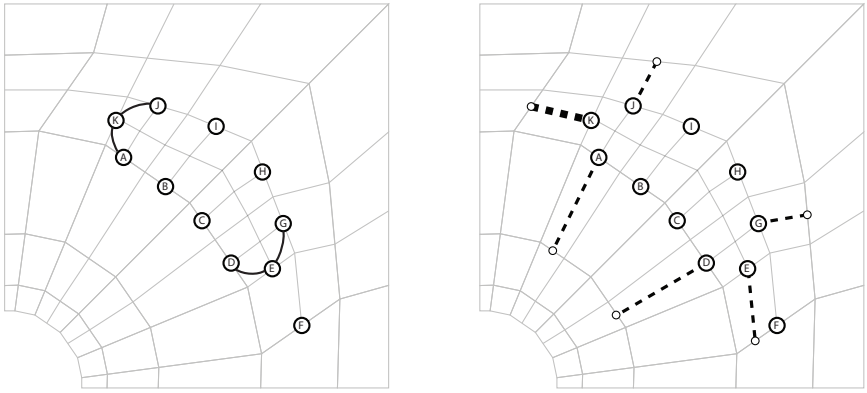


Figure 5.7: Initialization of a simple analysis-suitable local refinement example. (a) The initial analysis-suitable T-mesh before refinement. (b) Four T-mesh elements are subdivided to form T^2. (c) The coupled extended T-mesh $\mathsf{T}^{1\to 2}_{ext}$ corresponding to (a) and (b). (d) The coupled extension graph $E(\mathsf{T}^{1\to 2}_{ext})$. The graph is not empty so nesting does not necessarily hold.

We now begin to add insertion edges to T^2 until $E(\mathsf{T}^{1\to 2}_{ext})$ is empty. The result of the first iteration of the algorithm is shown in Figure 5.8. The input coupled extension graph is shown in Figure 5.8a. The corresponding set of insertion edges is shown in Figure 5.8b. Notice that insertion edges are only created for T-junctions where the weight of the corresponding node in $E(\mathsf{T}^{1\to 2}_{ext})$ is nonzero. Each insertion edge is then added to T^2 as a T-mesh edge and the change in graph weight ΔW and total resulting graph weight W are computed as shown in the Table of Figure 5.8. The shaded cells are ΔWs which must be computed during this iteration. Since this is the first iteration all must be computed. The insertion edge with the largest ΔW is then selected and inserted into the T-mesh. For this iteration, insertion edge K (the bold dashed line in Figure 5.8b) is inserted into T^2. Notice that in this case insertion edge E could also have been selected. Both have a ΔW of 2.

The second and final iteration of the algorithm is shown in Figure 5.9. The coupled extension graph is shown in Figure 5.9a. Notice the presence of the new T-mesh edge in T^2 which corresponds to insertion edge K from the previous step. The current insertion edges are shown in Figure 5.9b. There are now only three since the previous iteration decoupled nodes A, J, and K. The current values of ΔW and W are shown in the Table of Figure 5.9. None of the ΔW cells are shaded which indicates that the values are saved from a previous iteration and not computed. In general, only a small number of ΔWs need to be computed during each step of local refinement. The insertion edge with the largest ΔW is E which, when inserted, drives the total weight of the graph to zero thus terminating the refinement process. The final refined T-mesh is shown in Figure 5.10. The new edges are the dashed lines. If desired, the refinement operator $\mathbf{M}$ can be computed as described in Section 5.7.

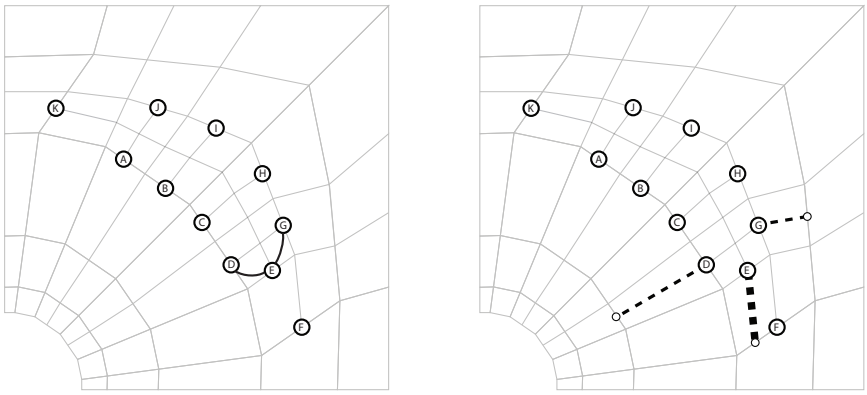


(a) Input graph (weight $= 4$)

(b) Insertion edges

	Insertion edge					
	A	D	E	G	J	K
ΔW	1	1	2	1	1	2
W	3	3	2	3	3	2

Figure 5.8: The first local refinement iteration for the example in Figure 5.7. (a) The input coupled extension graph. This graph has a weight of 4. (b) The corresponding insertion edges. ΔW and W for each insertion edge are shown in the Table. ΔW measures the change in graph weight after the edge is inserted into T^2. W is the total graph weight after the edge is inserted into T^2. The shaded cells indicate ΔW values which were computed during this iteration. Insertion edge K minimizes the graph weight and is inserted into the T-mesh as a T-mesh edge. Notice that in this case insertion edge E could also have been selected. Both have a ΔW of 2.



(a) Input graph (weight $= 2$)

(b) Insertion edges

	Insertion edge		
	D	E	F
ΔW	1	2	1
W	1	0	1

Figure 5.9: The second and final local refinement iteration for the example in Figure 5.7. (a) The input coupled extension graph. This graph has a weight of 2. (b) The corresponding insertion edges. ΔW and W for each insertion edge are shown in the Table. ΔW measures the change in graph weight after the edge is inserted into T^2. W is the total graph weight after the edge is inserted into T^2. Since no ΔW cells are shaded all values are saved from the previous refinement step (see Figure 5.8.) Insertion edge E drives the graph weight to zero and is inserted into the T-mesh as a T-mesh edge to complete the refinement process.

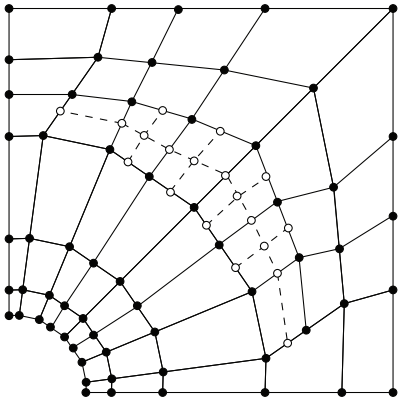


Figure 5.10: The final refined T-mesh for the example in Figure 5.7. The new edges and vertices are the dashed lines and open circles, respectively.

5.7 Computing the refinement operator $\mathbf{M}$

We now describe how the elements, $m_{A,B}$, of a refinement operator, $\mathbf{M}$, are computed. We recall that Steps 1 through 4 of the analysis-suitable local refinement algorithm in Section 5.6.3 guarantee that $\mathcal{T}_s^1 \subseteq \mathcal{T}_s^2$ and the existence of $\mathbf{M}$. We note that all knot comparisons between basis functions are made in a common knot coordinate system defined in the index space of T_s^2 (see Section 3.2.2.)

Before proceeding, we define several important index space concepts. For a basis function, $N_A^k \in \mathcal{T}_s^k$, we can construct the *index vectors*, $\mathbf{\Gamma}_{A,k} = \{\Gamma_{A,k}^i\}_{i=1}^2$, where $\Gamma_{A,k}^i = \{\tau_{A,1}^{k,i}, \tau_{A,2}^{k,i}, \ldots, \tau_{A,5}^{k,i}\}$ and $\tau_{A,j}^{k,i}$ is the index of $\xi_{A,j}^i$ in the index space of T^k. Using $\mathbf{\Gamma}_{A,k}$ we can then define the *index domain* $\Omega_{A,k}^I \subset \mathbb{R}^2$ as

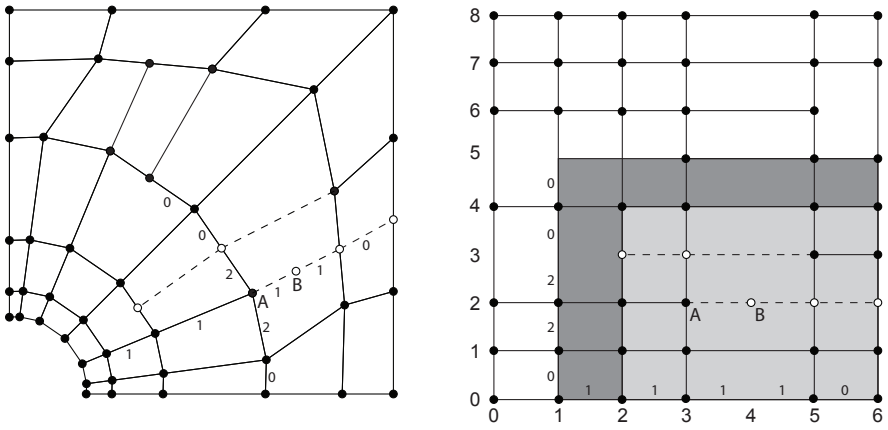
$$\Omega_{A,k}^I = \bigotimes_{i=1}^{2} \Omega_{A,k}^{I,i}, \tag{5.8}$$

where $\Omega_{A,k}^{I,i} = [\tau_{A,1}^{k,i}, \tau_{A,5}^{k,i}] \subset \mathbb{R}$.

We first initialize all entries in $\mathbf{M}$ to zero. Then, for each $N_A^1 \in \mathcal{T}_s^1$ and $N_B^2 \in \mathcal{T}_s^2$ such that $\Omega_{B,2}^I \subseteq \Omega_{A,2}^I$, we insert $\xi_{B,j}^i$ into Ξ_A^i if $\tau_{B,j}^{2,i} \notin \Gamma_{A,2}^i$, $i = 1, 2$, $j = 1, 2, \ldots, 5$. This constructs refined local knot vectors, $\bar{\mathbf{\Xi}}_A = \{\bar{\Xi}_A^i\}_{i=1}^2$. We then apply basis function refinement (see Section 5.6.1.1) using $\bar{\mathbf{\Xi}}_A$ to generate the linear combination

$$N_A^1 = \sum_{k=1}^{m} c_{A,k} N_k. \tag{5.9}$$

If there exists an N_k such that $N_k \equiv N_B^2$, then we set $m_{A,B} = c_{A,k}$. Note that the index vectors and domains for N_A^1 are constructed using the index space of T_s^2. This is possible since $\mathsf{T}_s^1 \subseteq \mathsf{T}_s^2$.



(a) (b)

Figure 5.11: Computing the refinement coefficient, $m_{A,B}$, corresponding to the exact representation of $N_A^1 \in \mathcal{T}_s^1$ in terms of $N_B^2 \in \mathcal{T}_s^2$. (a) The T-mesh, T_s^1, consists of the solid circles and lines. The topology (control points and edges) added during the topology phase of analysis-suitable local refinement (Steps 1 through 4 in Section 5.6.3) are denoted by the open circles and dashed lines, respectively. The topology phase ensures that $\mathcal{T}_s^1 \subseteq \mathcal{T}_s^2$. (b) The index space of T_s^2. The index domain $\Omega_{A,2}^I = [1,6] \otimes [0,5]$ is the union of the dark and lightly shaded rectangles and the index domain $\Omega_{B,2}^I = [2,6] \otimes [0,4]$ is the lightly shaded rectangle. The origin of the common knot coordinate system is $(1,0)$ in the index space. The knot intervals for the common knot coordinate system are shown in (a) and (b).

To illustrate, Figure 5.11a shows a T-mesh, T_s^1, where additional vertices and edges (open circles and dashed lines) have been added during the first four steps of the local refinement algorithm in Section 5.6.3 to form T_s^2. As a result $\mathcal{T}_s^1 \subseteq \mathcal{T}_s^2$. The knot intervals used in this example are shown next to the corresponding edges. The index space representation is shown in Figure 5.11b. For simplicity we choose a *global* knot coordinate system with an origin at $(1, 0)$ in the index space of T_s^2.

We now compute $m_{A,B}$, where the basis functions $N_A^1 \in \mathcal{T}_s^1$ and $N_B^2 \in \mathcal{T}_s^2$ are associated with the vertices labeled A and B in Figure 5.11. In this case, the index vectors, with respect to the index space of T_s^2, are

$$\mathbf{\Gamma}_{A,2} = \begin{bmatrix} 1, 2, 3, 5, 6 \\ 0, 1, 2, 4, 5 \end{bmatrix} \tag{5.10}$$

and

$$\mathbf{\Gamma}_{B,2} = \begin{bmatrix} 2, 3, 4, 5, 6 \\ 0, 1, 2, 3, 4 \end{bmatrix}. \tag{5.11}$$

The index domain, $\Omega_{A,2}^I = [1, 6] \otimes [0, 5]$, is the union of the dark and lightly shaded rectangles in Figure 5.11b and the index domain, $\Omega_{B,2}^I = [2, 6] \otimes [0, 4]$, is the lightly shaded rectangle.

In this coordinate system, the local knot vectors for N_A^1 and N_B^2 are

$$\mathbf{\Xi}_A = \begin{bmatrix} 0, 1, 2, 4, 4 \\ 0, 0, 2, 4, 4 \end{bmatrix} \tag{5.12}$$

and

$$\mathbf{\Xi}_B = \begin{bmatrix} 1, 2, 3, 4, 4 \\ 0, 0, 2, 4, 4 \end{bmatrix}. \tag{5.13}$$

Obviously, $\Omega_{B,2}^I \subseteq \Omega_{A,2}^I$, and since $\tau_{B,3}^{2,1} = 4$ and $\tau_{B,4}^{2,2} = 3$ are not in $\Gamma_{A,2}^1$ and $\Gamma_{A,2}^2$ we insert the corresponding knots $\xi_{B,3}^1 = 3$ and $\xi_{B,4}^2 = 4$ into Ξ_A^1 and Ξ_A^2. This

results in the refined local knot vectors

$$\bar{\boldsymbol{\Xi}}_A = \begin{bmatrix} 0,1,2,3,4,4 \\ 0,0,2,4,4,4 \end{bmatrix}. \tag{5.14}$$

Applying the refinement equations, (5.3) - (5.5), to $\bar{\boldsymbol{\Xi}}_A$, in each univariate direction, and taking the tensor product of the resulting coefficients results in

$$N_A^1 = c_{A,1}N_1 + c_{A,2}N_2 + c_{A,3}N_3 + c_{A,4}N_4 \tag{5.15}$$

$$= \frac{3}{4}N_1 + \frac{1}{4}N_3 \tag{5.16}$$

where

$$\boldsymbol{\Xi}_1 = \begin{bmatrix} 0,1,2,3,4 \\ 0,0,2,4,4 \end{bmatrix} \tag{5.17}$$

and

$$\boldsymbol{\Xi}_3 = \begin{bmatrix} 1,2,3,4,4 \\ 0,0,2,4,4 \end{bmatrix}. \tag{5.18}$$

Since $N_3 \equiv N_B^2$, we have that $m_{A,B} = \frac{1}{4}$.

5.8 Applying analysis-suitable local refinement

We now explore the application and behavior of analysis-suitable T-splines and local refinement. In this example, we use analysis-suitable local refinement to transform an initial coarse T-spline ship hull design into an analysis-suitable model. The analysis-suitable model can then be used directly in isogeometric analysis by way of Bézier extraction as described in Chapter 4. We note that this same approach can also be used as an adaptive finite element solution strategy. A demanding application of adaptive T-spline local refinement in the context of a phase-field fracture model is described in Chapter 7.4.

In Figure 5.12, we schematically illustrate the process used to perform analysis-suitable local refinement. First, a set of Bézier elements, generated using Bézier

extraction, is flagged by the user or finite element solver. Next, the Bézier elements are used to identify corresponding T-mesh elements. We recall that several Bézier elements may correspond to a single T-mesh element as described in Section 3.4.5 and shown in Figure 5.12 on the left. The selected T-mesh elements are then refined to generate T^2 as described in Section 5.6.3. In this case, the T-mesh elements are simply subdivided. Once the selected T-mesh elements are subdivided, Steps 2 through 5 of the local refinement algorithm presented in Section 5.6.3 are applied. This generates the final refined analysis-suitable T-spline space. Bézier extraction is then performed resulting in a new set of Bézier elements. This process is then repeated until the resolution of the T-spline is sufficient for the application.

The T-spline container ship hull in Figure 5.13 is first designed using the T-spline plugin for Rhino3d [114]. It is then transformed into an analysis-suitable model using local refinement. T-splines are a popular technology in ship hull design because an entire hull can be modeled by a single watertight surface with a minimal number of control points [102]. In analysis, however, far more degrees-of-freedom are often required to capture the physical phenomenon of interest. In other words, the initial T-mesh must undergo additional refinements to create models that satisfy the needs of analysis.

To demonstrate the pertinent ideas, we assume that the final analysis-suitable model of the hull must be sufficiently resolved to capture the response of the ship in the two regions outlined in Figure 5.14. This will require refinements in the region of the hull corresponding to the rectangle followed by highly localized refinements along the region corresponding to the curve. Six iterations of refinement are performed as shown in Figures 5.15 through 5.20. The initial T-spline of the hull contains just 75 control points and 36 Bézier elements.

The first iteration of local refinement of the ship hull is shown in Figure 5.15.

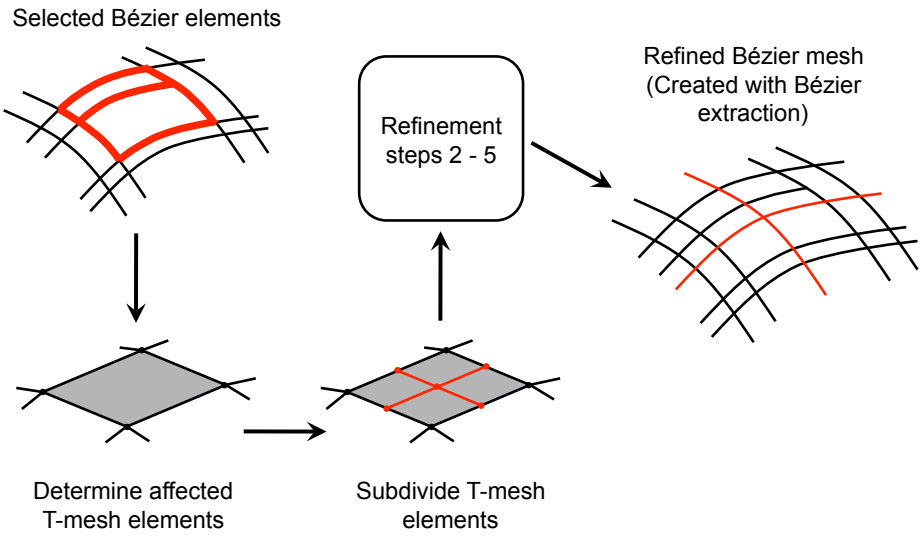


Figure 5.12: An analysis-suitable local refinement framework.

A set of Bézier elements is selected for refinement as shown on the top of Figure 5.15. The refinement framework described in Figure 5.12 is then applied. First, the selected Bézier elements are used to select corresponding T-mesh elements. These T-mesh elements are subdivided and the local refinement algorithm described in Section 5.6.3 is applied to the resulting subdivided T-mesh. The control points added during local refinement are shown in the middle of Figure 5.15. Notice that these control points remain localized to the region of selected Bézier elements. The refined control mesh is shown on the bottom of Figure 5.15 with the new control points highlighted. The refined set of Bézier elements is then extracted from the re-

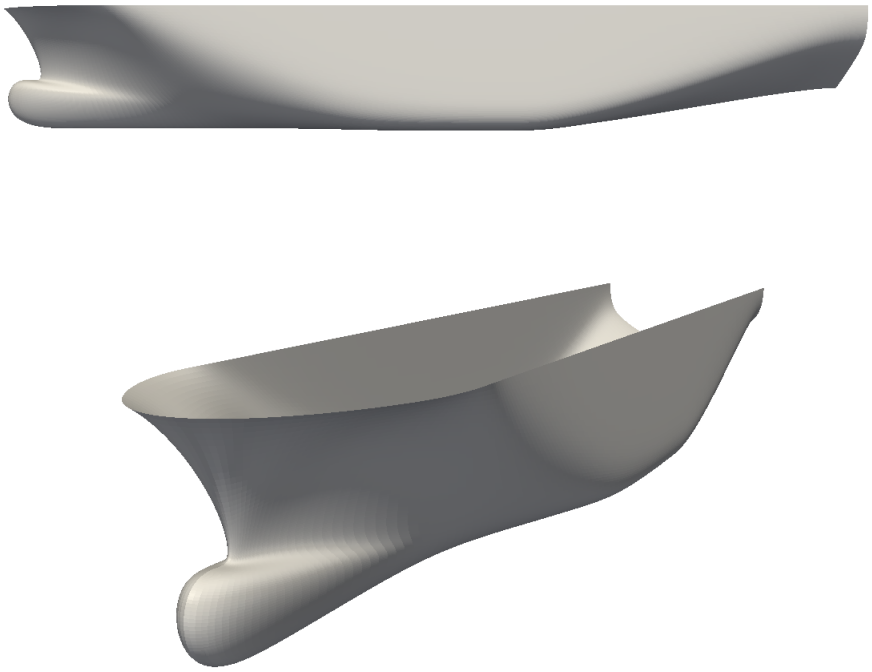


Figure 5.13: A T-spline container ship hull. The surface is C^2-continuous everywhere.

fined T-mesh as shown in Figure 5.16 on the top. We note that the transpose of the refinement operator, $\mathbf{M}^T$ (see Sections 5.6.1.2 and 5.7), is used to update control point positions after each refinement step. This ensures that the geometry and its parameterization are preserved exactly.

The next five iterations of local refinement are shown in Figures 5.16 through 5.20. In Figure 5.16 and Figure 5.17 the rectangular region in Figure 5.14 undergoes additional local refinement. In Figures 5.18 through 5.20 highly localized refinement is performed along the curve in Figure 5.14. Notice that the refinement pattern fol-

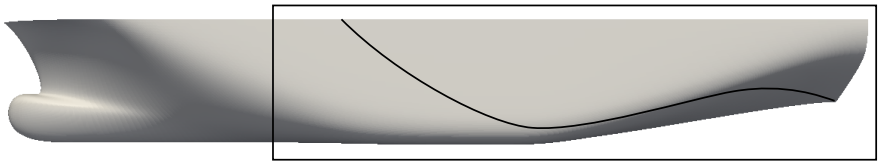


Figure 5.14: The regions of the container ship hull where analysis-suitable local refinement will be performed. First, refinement will be performed in the rectangular region followed by highly localized refinement along the curve.

lows the curve without excessive propagation of control points while preserving the C^2-continuous analysis-suitable T-spline basis. The fully resolved analysis-suitable model has 1722 control points and 1922 Bézier elements. The final Bézier mesh is shown in Figure 5.20 on the bottom and the final T-mesh is shown immediately above it. The sequence of C^2-continuous T-spline spaces is nested and the initial geometry is exactly preserved throughout.

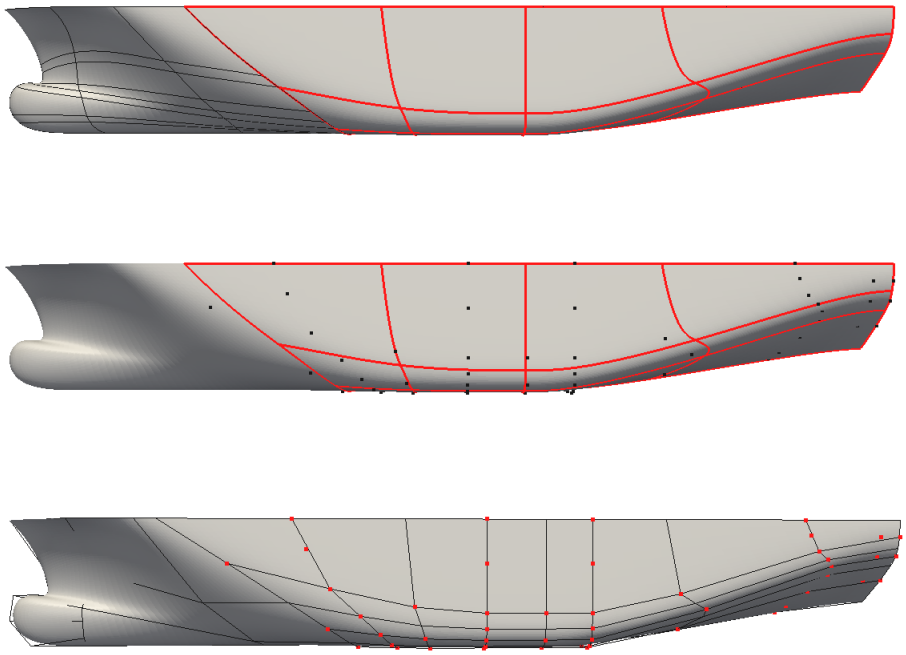


Figure 5.15: The first iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.)

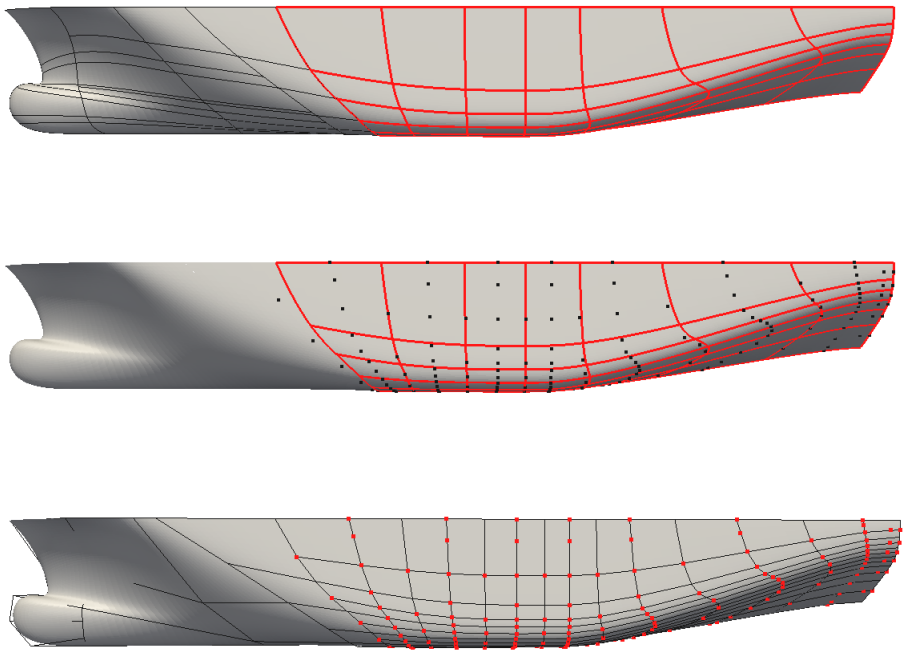


Figure 5.16: The second iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.)

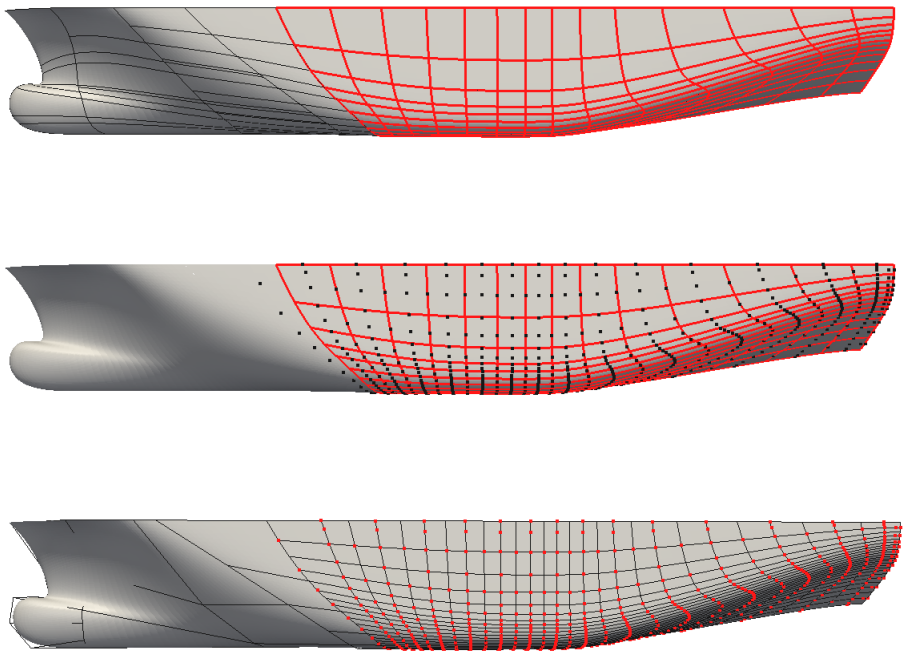


Figure 5.17: The third iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.)

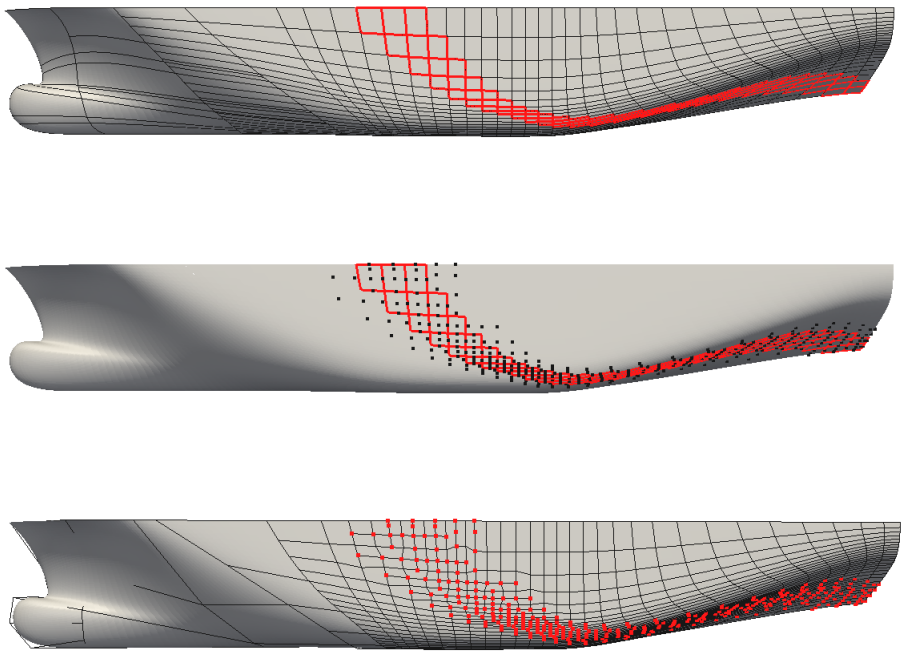


Figure 5.18: The fourth iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. Notice that analysis-suitable local refinement remains localized to the Bézier elements selected for refinement. (Note: Some of the T-mesh edges are hidden behind the ship hull.)

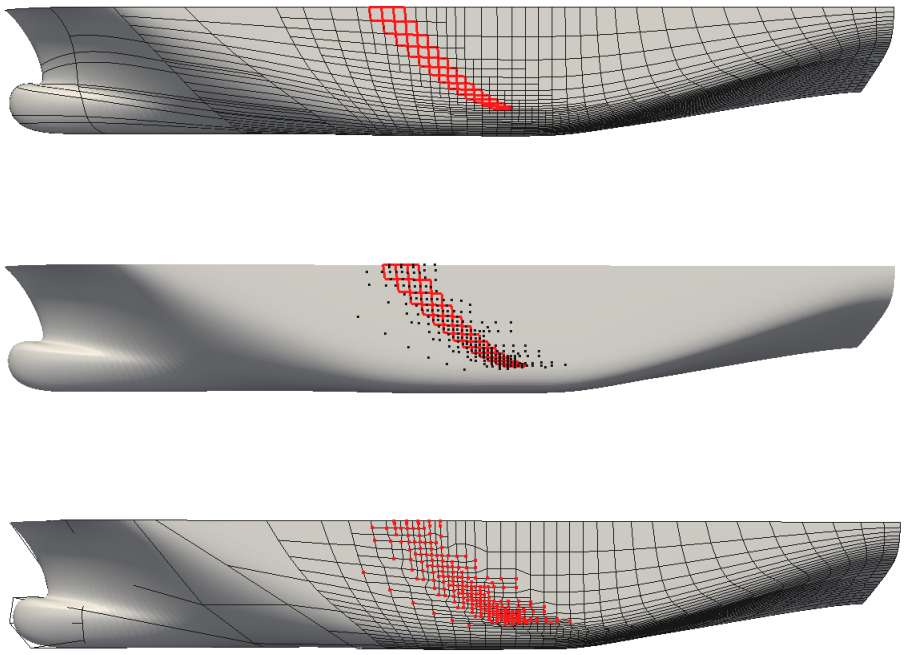


Figure 5.19: The fifth iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown in the middle. The refined T-mesh is shown on the bottom. The new control points are highlighted in red. Notice that analysis-suitable local refinement remains localized to the Bézier elements selected for refinement. (Note: Some of the T-mesh edges are hidden behind the ship hull.)

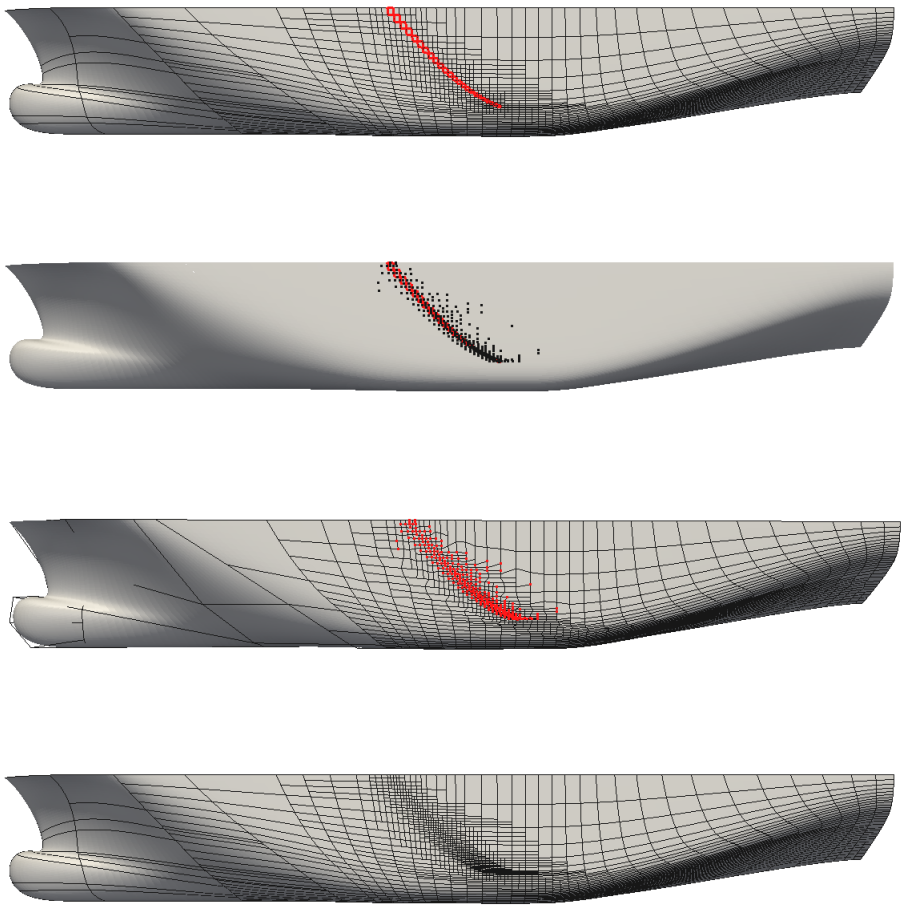


Figure 5.20: The sixth and final iteration of analysis-suitable local refinement of the container ship hull in Figure 5.13. The Bézier elements are shown on the top with those elements selected for refinement highlighted in red. The control points added during analysis-suitable local refinement are shown below that. The refined final T-mesh is then shown. The new control points are highlighted in red. (Note: Some of the T-mesh edges are hidden behind the ship hull.) The final Bézier element mesh is shown on the bottom. The refinements form a nested sequence of C^2-continuous spline spaces. The geometry of the hull is unchanged during the refinements.

5.9 Linear independence

While linear independence is not required (although desirable) for most CAGD applications, it is imperative for isogeometric analysis. Linear dependencies are never allowed in practice in a finite element basis, since the equation system is not invertible, and squaring the system (as is done in a least squares approach) does not remove rank deficiency. While redundancies can be removed and the reduced squared system formed, this is a risky approach since the condition number is also squared in the process. Consequently, techniques like this were purged from the FEA literature 50 years ago. Thus, the analysis community requires bases that are assured *a priori* to be linearly independent.

We now provide a sketch of the proof that the set of blending functions underlying analysis-suitable T-splines form a basis. For the extremely technical details we refer the interested reader to [71].

We note that for strictly positive weights (assumed in practice), linear independence of the polynomial blending functions is equivalent to linear independence of the *rational* blending functions. Thus, the choice of (strictly positive) weights does not affect linear independence of the rational blending functions.

5.9.1 The T-spline-to-NURBS transform matrix, $\mathbf{N}$

Given any T-spline, we can compute the *T-spline-to-NURBS transform matrix*, $\mathbf{N}$, by extending all T-junctions to the boundary of the T-mesh and applying the local refinement algorithm described in Chapter 5.6. This results in a refinement operator, $\mathbf{M}$. The transform matrix is simply the transpose of this refinement operator. In other words, $\mathbf{N} = \mathbf{M}^T$. Notice that the refined T-mesh is a NURBS whose blending functions form a basis.

Theorem 5.9.1. *The T-spline-to-NURBS refinement operator,* $\mathbf{N}$*, for any T-spline surface is unique. Furthermore, a necessary and sufficient condition for a T-spline's blending functions to be linearly independent is that* $\mathbf{N}$ *is full rank.*

Proof. Since the refined space is a NURBS space, each column of $\mathbf{N}$ is unique, and so $\mathbf{N}$ itself is unique. By definition, T-spline blending functions are linearly independent if and only if there do not exist constants, c_A, not all zero, such that

$$c_A N_A^1 + \ldots + c_{n_2} N_{n_2}^1 = (c_A, \ldots, c_{n_2}) \begin{pmatrix} N_A^1 \\ \vdots \\ N_{n_2}^1 \end{pmatrix} = 0. \tag{5.19}$$

This means that linear independence requires

$$(c_A, \ldots, c_{n_2}) \begin{pmatrix} N_A^1 \\ \vdots \\ N_{n_2}^1 \end{pmatrix} = (c_A, \ldots, c_{n_2}) \mathbf{N}^T \begin{pmatrix} N_A^2 \\ \vdots \\ N_{n_1}^2 \end{pmatrix} = 0. \tag{5.20}$$

Since $\{N_A^2\}$ is a basis, the necessary and sufficient condition for linear dependence of the T-spline blending functions becomes

$$(c_A, \ldots, c_{n_2}) \mathbf{N}^T = \mathbf{N} \begin{pmatrix} N_A^1 \\ \vdots \\ N_{n_2}^1 \end{pmatrix} = 0 \tag{5.21}$$

for c_A not all zero. But this will only happen if $\mathbf{N}$ is not full rank. □

5.9.2 Column reduction

Recall that the dimension of the null space of a matrix is called its nullity and that for an $n_1 \times n_2$ matrix $\mathbf{N}$, the Rank-nullity theorem states that the nullity of $\mathbf{N}$ is $n_2 - \operatorname{rank} \mathbf{N}$. Thus, linear independence of T-spline blending functions is equivalent to the nullity of $\mathbf{N}$ being zero.

If all elements of row B of $\mathbf{N}$ are zero except m_{BA}, column A is called an *innocuous* column. *Column reduction* is the operation of removing an innocuous column from $\mathbf{N}$ along with any zero rows that the column removal may have introduced. Denote by $\tilde{\mathbf{N}}$ the matrix that results after performing all possible column reductions.

Lemma 5.9.2. *Nullity of an $n_1 \times n_2$ matrix with $n_1 \geq n_2$ is invariant under column reduction, so $\mathbf{N}$ and $\tilde{\mathbf{N}}$ have the same nullity.*

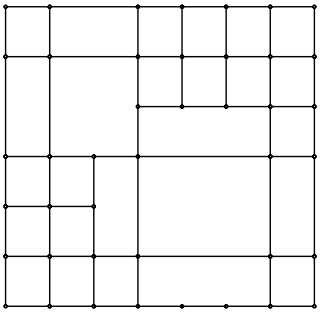
Proof. Follows from the Rank-nullity theorem. □

Corollary 5.9.3. *If $\tilde{\mathbf{N}} = \emptyset$, the T-spline's blending functions are linearly independent for all global knot vectors.*

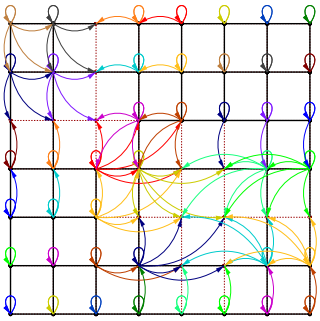
5.9.3 The influence graph, G

We can visualize column reduction using a directed graph G, drawn on a T-mesh, that we call an *influence graph*. G contains two types of nodes: *T-nodes* correspond to T-spline control points, and *N-nodes* correspond to underlying NURBS control points. Edges in G originate at T-nodes and terminate at N-nodes and represent non-zero elements of $\mathbf{N}$: If m_{AB} is non-zero, an edge is drawn from T-node B to N-node A. A T-node and N-node that have the same index coordinates are said to correspond to each other, and every T-node points to its corresponding N-node. Figure 5.21b shows the influence graph G for the T-mesh in Figure 5.21a.

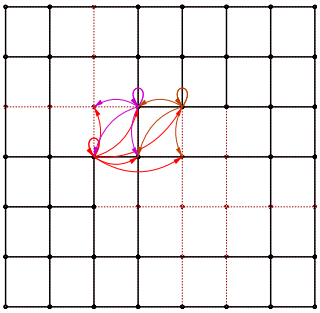
The valence of an N-node is the number of edges that point to it. The valence of a T-node is the number of edges originating from it. An innocuous node is any T-node that points to at least one N-node of valence one, and represents an innocuous column of $\mathbf{N}$. Pruning a graph is the graphical equivalent of column reduction, and



(a) A T-mesh, T



(b) The influence graph, G



(c) A V2-subgraph

Figure 5.21: A T-mesh and its influence graph

consists of eliminating an innocuous node, edges originating from it, and any N-nodes that no longer are pointed to. A graph from which all innocuous nodes have been pruned is said to be fully pruned. Figure 5.21c shows the fully pruned version of Figure 5.21b.

A subgraph of G consists of any set of T-nodes, all N-nodes pointed to by those T-nodes, and all edges connecting those nodes. A *V2-subgraph* is a subgraph whose N-nodes all have a valence of at least two. A fully pruned graph is either empty, or consists of one or more V2-subgraphs. Figure 5.21c shows a V2-subgraph with three T-nodes and six N-nodes.

Theorem 5.9.4. *If the influence graph for a T-mesh contains no V2-subgraphs, the T-spline has linearly independent blending functions.*

Proof. See Corollary 5.9.3, noting that pruning visualizes column reduction. □

Theorem 5.9.5. *Analysis-suitable T-spline surfaces have linearly independent blending functions.*

Proof. This result is established by showing that analysis-suitable T-meshes do not produce any V2-subgraphs. This requires a sophisticated abstraction of T-mesh topological quantities. See [71], Theorem 7 for the details. □

Chapter 6

Extraordinary points

When generating real-world T-spline models, the presence of extraordinary points in the T-mesh is inevitable. For vertices which are not on the boundary of the T-mesh, an *extraordinary point* is a vertex which is not a T-junction and whose *valence* or the number of edges touching the vertex is not four. Figure 6.1 shows a T-mesh with two extraordinary points denoted by open red circles. Generating efficient and simple element technology near extraordinary points which meet the needs of *both* design and analysis is crucial if isogeometric analysis is to be competitive with traditional finite element discretization schemes.

The treatment and analysis of extraordinary points in control meshes that have quadrilateral faces has a rich history in CAGD. A popular approach in CAGD to dealing with such extraordinary points is to use subdivision surfaces [30, 31, 43, 74, 93, 94, 98, 112]. In practice, the most popular approaches generalize uniform B-spline knot insertion or h-refinement with notable exceptions being NURSS [107] and T-NURCCs [106], which handle non-uniform knots and T-junctions, respectively. Unfortunately, in all cases, the elements near the extraordinary point comprise an infinite sequence of piecewise polynomials. This complication hampers their utility as an analysis technology from both a practical and theoretical point-of-view. See Appendix C for a thorough discussion of subdivision-based element forms and the inherent difficulties. We mention in passing that certain subdivision schemes have been used as a basis for finite element analysis [29, 32–35].

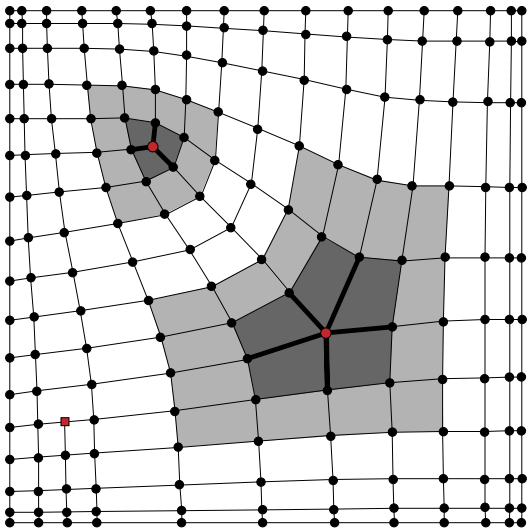


Figure 6.1: A T-mesh of arbitrary topology. Extraordinary points are denoted by red hollow circles and T-junctions are denoted by red hollow squares. The one-ring neighborhoods are composed of the darkly shaded elements and the two-ring neighborhoods are composed of the dark and lightly shaded elements. The spoke edges are denoted by the thick black lines.

As an alternative approach to dealing with extraordinary points, the CAGD community has explored *finite* polynomial representations for the surface surrounding an extraordinary point. These approaches [22, 42, 73, 75–77, 87, 109, 122] are more amenable to real-time applications, modern GPU processors, and high-end engineering design where curvature continuity is required.

In like manner, we formulate a simple isogeometric element for analysis-suitable T-splines near extraordinary points which does not depend upon subdivision. We handle *non-uniform* knots and show that the element form fits into the Bézier extraction paradigm described in Chapter 4. Thus, Bézier extraction provides a unified framework for *all* element types in a T-spline, regardless of proximity to an extraordinary point or topological complexity.

6.1 The T-mesh with extraordinary points

Most T-meshes of practical importance contain extraordinary points. We denote the valence of an extraordinary vertex by N. In a T-mesh with extraordinaty points the *spoke edges* are the T-mesh edges which touch an extraordinary point. The T-mesh elements which touch the extraordinary point form the *one-ring neighbhorhood* of the vertex. The *two-ring neighborhood* is the one-ring neighborhood and the elements which touch the one-ring neighbhorhood. An *n-ring neighborhood* can be formed iteratively in the obvious way. We call the T-mesh elements which compose a two-ring neighborhood *extraordinary elements*. In Figure 6.1 the hollow red circle in the upper left is a valence three extraordinary point and the hollow red circle in the lower right is valence five extraordinary point. The spoke edges are denoted by the bold black lines. The one-ring neighborhoods are composed of the darkly shaded elements surrounding each extraordinary point and the two-ring neighborhoods are composed of the union of the dark and lightly shaded

T-mesh elements. We call the T-mesh after all two-ring neighborhoods are removed the *regular T-mesh*.

6.2 The T-spline basis near extraordinary points

Once a valid knot interval configuration is assigned to a T-mesh (see Section 3.1), a T-spline basis can be constructed. For every vertex in the T-mesh, a T-spline basis function is constructed. For T-splines with extraordinary points, basis function construction and representation can be standardized using Bézier extraction and element extraction operators. Bézier extraction, as described in Chapter 4, is used to construct the Bézier elements in the regular T-mesh. For each two-ring neighborhood, T-spline basis functions are defined using a two-step Bézier extraction procedure:

Step 1 Generalized Bézier extraction is applied to the two-ring neighborhood.

Step 2 The resulting extraction operators are perturbed to construct a G^1-continuous basis.

6.2.1 Generalized Bézier extraction

We can generalize bicubic Bézier extraction to handle extraordinary elements. As shown in Figure 6.2a, the 16 Bézier control points defining each extraordinary element can be partitioned into 4 *face points*, $\mathbf{Q}_6^f$, $\mathbf{Q}_7^f$, $\mathbf{Q}_{10}^f$, $\mathbf{Q}_{11}^f$, 8 *edge points*, $\mathbf{Q}_2^e$, $\mathbf{Q}_3^e$, $\mathbf{Q}_5^e$, $\mathbf{Q}_8^e$, $\mathbf{Q}_9^e$, $\mathbf{Q}_{12}^e$, $\mathbf{Q}_{14}^e$, and $\mathbf{Q}_{15}^e$, and 4 *vertex points*, $\mathbf{Q}_1^e$, $\mathbf{Q}_4^e$, $\mathbf{Q}_{13}^e$, and $\mathbf{Q}_{16}^e$. Notice that each of the four corner T-spline control points (denoted by solid circles) may or may not be extraordinary vertices.

Referring to Figure 6.2b, the linear combinations defining the four face

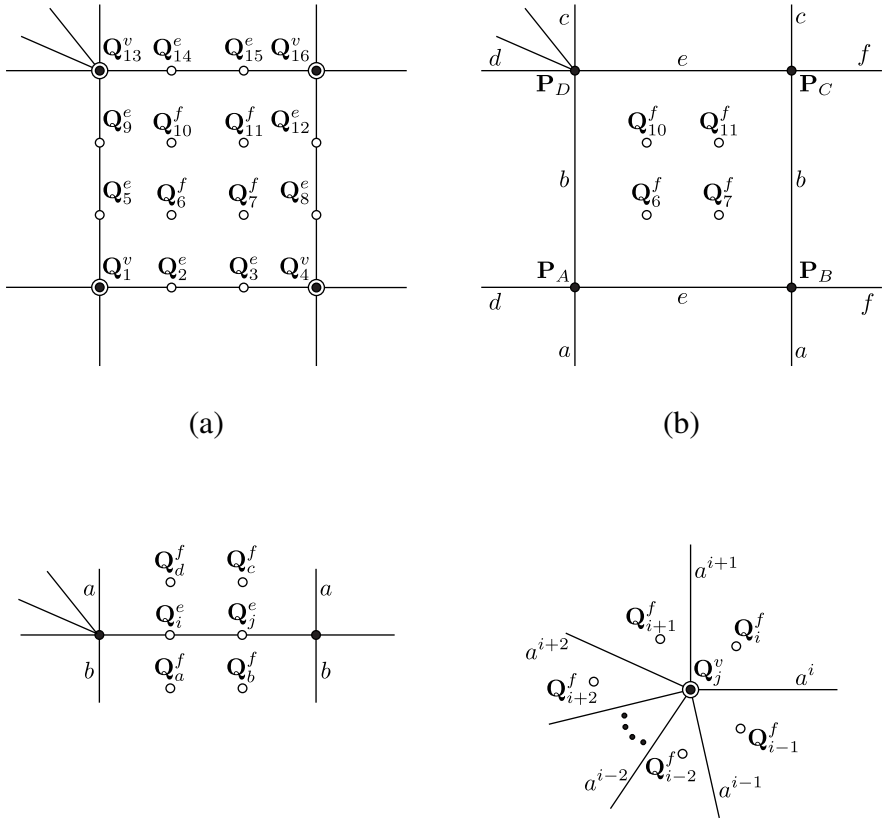


Figure 6.2: Generalized bicubic Bézier extraction. (a) Face, edge, and vertex points defining an extraordinary element. The Bézier control points are denoted by open circles while the original T-spline control points are denoted by solid circles. Face Bézier points are denoted by a superscript f, edge Bézier points are denoted by a superscript e, and vertex Bézier points are denoted by a superscript v. Notice that the T-spline control points may or may not be extraordinary points. (b) Face points for a Bézier element. Each face point is written in terms of T-spline control points $\mathbf{P}_A$ through $\mathbf{P}_D$ and knot intervals a through f. (c) Edge points corresponding to an edge of an extraordinary element. Each edge point is written in terms of Bézier face points of neighboring Bézier elements and knot intervals a and b. (d) A vertex point corresponding to a corner of an extraordinary element. Each vertex point is written in terms of all adjacent face points and spoke edge knot intervals.

points are

$$\mathbf{Q}_6^f = \left(\frac{b+c}{a+b+c}\right)\left(\frac{e+f}{d+e+f}\right)\mathbf{P}_A + \left(\frac{b+c}{a+b+c}\right)\left(\frac{d}{d+e+f}\right)\mathbf{P}_B \quad (6.1)$$
$$+ \left(\frac{a}{a+b+c}\right)\left(\frac{d}{d+e+f}\right)\mathbf{P}_C + \left(\frac{a}{a+b+c}\right)\left(\frac{e+f}{d+e+f}\right)\mathbf{P}_D,$$

$$\mathbf{Q}_7^f = \left(\frac{b+c}{a+b+c}\right)\left(\frac{f}{d+e+f}\right)\mathbf{P}_A + \left(\frac{b+c}{a+b+c}\right)\left(\frac{d+e}{d+e+f}\right)\mathbf{P}_B \quad (6.2)$$
$$+ \left(\frac{a}{a+b+c}\right)\left(\frac{d+e}{d+e+f}\right)\mathbf{P}_C + \left(\frac{a}{a+b+c}\right)\left(\frac{f}{d+e+f}\right)\mathbf{P}_D,$$

$$\mathbf{Q}_{10}^f = \left(\frac{c}{a+b+c}\right)\left(\frac{e+f}{d+e+f}\right)\mathbf{P}_A + \left(\frac{c}{a+b+c}\right)\left(\frac{d}{d+e+f}\right)\mathbf{P}_B \quad (6.3)$$
$$+ \left(\frac{a+b}{a+b+c}\right)\left(\frac{d}{d+e+f}\right)\mathbf{P}_C + \left(\frac{a+b}{a+b+c}\right)\left(\frac{e+f}{d+e+f}\right)\mathbf{P}_D,$$

and

$$\mathbf{Q}_{11}^f = \left(\frac{c}{a+b+c}\right)\left(\frac{f}{d+e+f}\right)\mathbf{P}_A + \left(\frac{c}{a+b+c}\right)\left(\frac{d+e}{d+e+f}\right)\mathbf{P}_B \quad (6.4)$$
$$+ \left(\frac{a+b}{a+b+c}\right)\left(\frac{d+e}{d+e+f}\right)\mathbf{P}_C + \left(\frac{a+b}{a+b+c}\right)\left(\frac{f}{d+e+f}\right)\mathbf{P}_D,$$

where bold upper-case letters denote control points and lower case letters denote knot intervals.

Referring to Figure 6.2c, any two edge points along an edge are defined as

$$\mathbf{Q}_i^e = \left(\frac{a}{a+b}\right)\mathbf{Q}_a^f + \left(\frac{b}{a+b}\right)\mathbf{Q}_d^f \quad (6.5)$$

and

$$\mathbf{Q}_j^e = \left(\frac{a}{a+b}\right)\mathbf{Q}_b^f + \left(\frac{b}{a+b}\right)\mathbf{Q}_c^f. \quad (6.6)$$

Finally, referring to Figure 6.2d, each vertex point is defined as

$$\mathbf{Q}_j^v = \sum_{i=1}^{N} \left(\frac{a^{i-1}}{a^{i-1} + a^{i+1}} \right) \left(\frac{a^{i+2}}{a^i + a^{i+2}} \right) \mathbf{Q}_i^f. \tag{6.7}$$

Notice that both edge and vertex points are defined in terms of face points of neighboring Bézier elements. If the initial control mesh is a tensor product mesh these rules are equivalent to repeated knot insertion as presented in [23, 99].

Generalized Bézier extraction defines the *transpose* of the element extraction operators, $\mathbf{C}^e$, for each extraordinary element. Figure 6.3, on the top, shows the T-spline basis functions, resulting from generalized Bézier extraction, corresponding to the two extraordinary points in Figure 6.1. The Bézier mesh defining each basis function is also plotted. Figure 6.3, on the bottom, shows the T-spline basis function, corresponding to the valence five extraordinary point in Figure 6.1. The lack of smoothness of the basis function is evident.

6.2.2 Smoothing the extraction

To enforce G^1 continuity along spoke edges a straightforward perturbation of extraordinary element extraction operators is performed. Our method requires *biquartic* one-ring extraordinary elements.

For each T-spline basis function, N_A, which is non-zero over at least two adjacent one-ring extraordinary elements, we compute the set of new extraction coefficients corresponding to a degree elevated biquartic extraordinary element. We note that degree elevation does not change N_A. Then, a sequential least-squares problem is solved. Given a fairing matrix $\mathbf{F}_A \in \mathbb{R}^{40N \times (1+20N)}$ and corresponding right-hand side $\mathbf{f}_A \in \mathbb{R}^{1+20N}$ and G^1 constraint matrix $\mathbf{G}_A \in \mathbb{R}^{(1+20N)\times(1+20N)}$ and corresponding right-hand side $\mathbf{g}_A \in \mathbb{R}^{1+20N}$ we want to find a vector of smoothed

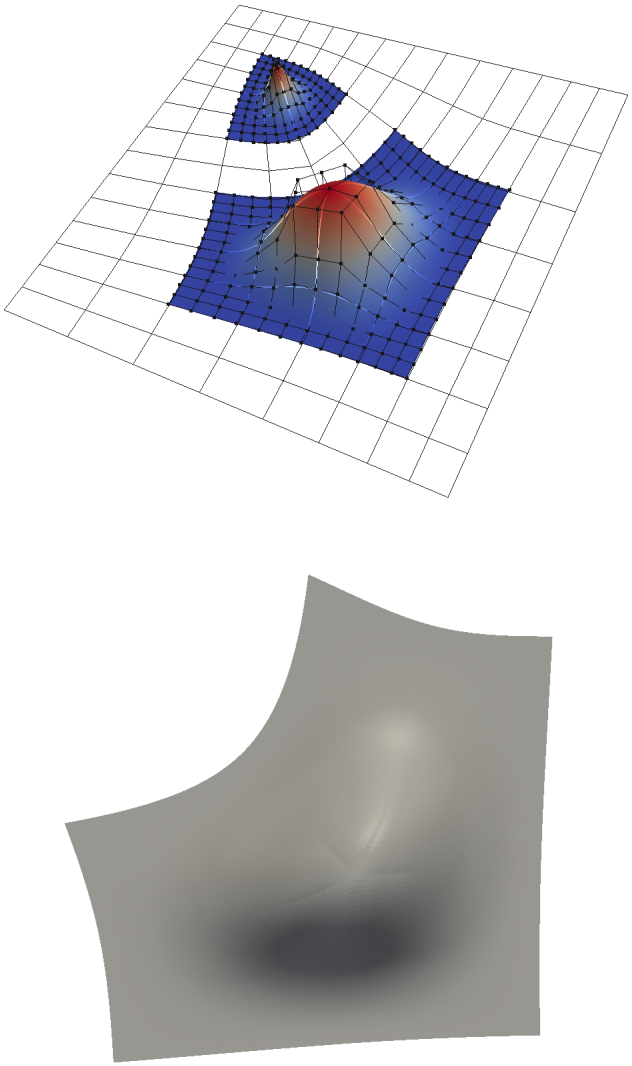


Figure 6.3: Generalized Bézier extraction. (top) The T-spline basis functions corresponding to the two extraordinary points in Figure 6.1. These basis functions *do not* lie in a portion of the T-mesh with a rectangular grid topology. The Bézier mesh defining each basis function is also plotted. (bottom) The T-spline basis function, corresponding to the valence five extraordinary point in Figure 6.1. The lack of smoothness of the basis function is evident.

extraction coefficients $\tilde{\mathbf{c}}_A \in \mathbb{R}^{1+20N}$ which solves

$$\min_{\tilde{\mathbf{c}}_A \in \mathbf{S}_A} ||\mathbf{F}_A \tilde{\mathbf{c}}_A - \mathbf{f}||_2 \tag{6.8}$$

where

$$\mathbf{S}_A = \{\tilde{\mathbf{c}}_A \,|\, ||\mathbf{G}_A \tilde{\mathbf{c}}_A - \mathbf{g}||_2 = \min\}\,. \tag{6.9}$$

Once $\mathbf{F}_A$, $\mathbf{f}_A$, $\mathbf{G}$, and $\mathbf{g}_A$ are assembled we solve the constrained least squares problem using the method of direct elimination as described in [21]. This approach correctly handles any linear dependencies which may exist in the constraint matrix, $\mathbf{G}_A$.

We note that to assemble the global equation system for a one-ring neighborhood of extraordinary elements, the local indices for an extraction coefficient, $c^{i,A}_{\alpha\beta}$, corresponding to the extraction of N_A over the i^{th} extraordinary element must be mapped to a global index I. This map can be written as,

$$I(i,\alpha,\beta) = \begin{cases} \alpha,\beta = 1, & 1, \\ \alpha = 1, \beta \neq 1, & 20i + \beta \\ \alpha \neq 1, & 20(i-1) + 4(\beta - 1) + \alpha. \end{cases} \tag{6.10}$$

Figure 6.4 shows the action of $I(i,\alpha,\beta)$ on the local indices of the extraction coefficients, $c^{i,A}_{\alpha\beta}$.

6.2.2.1 Assembling the constraint equations

As a result of generalized Bézier extraction, extraordinary element $\mathbf{w}^i$ is C^1-continuous with $\mathbf{x}^i$ and $\mathbf{z}^i$, $i = 1, \ldots, N$, as shown in Figure 6.5. To maintain C^1 continuity between these elements during smoothing only requires that the extraction coefficients remain the same. This is satisfied if the equations $\tilde{c}^A_{I(i,\alpha,\beta)} = c^{i,A}_{\alpha\beta}$ for $\alpha \geq 4$ and $\beta \geq 1$ or $2 \leq \alpha \leq 3$ and $\beta \geq 4$, $i = 1, \ldots, N$ are assembled into $\mathbf{G}_A$ and $\mathbf{g}_A$.

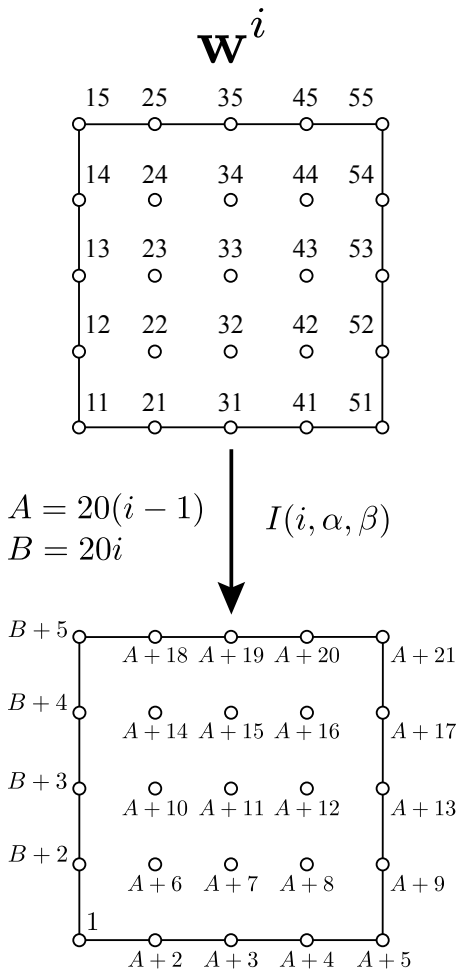


Figure 6.4: The action of the local-to-global index map, $I(i, \alpha, \beta)$.

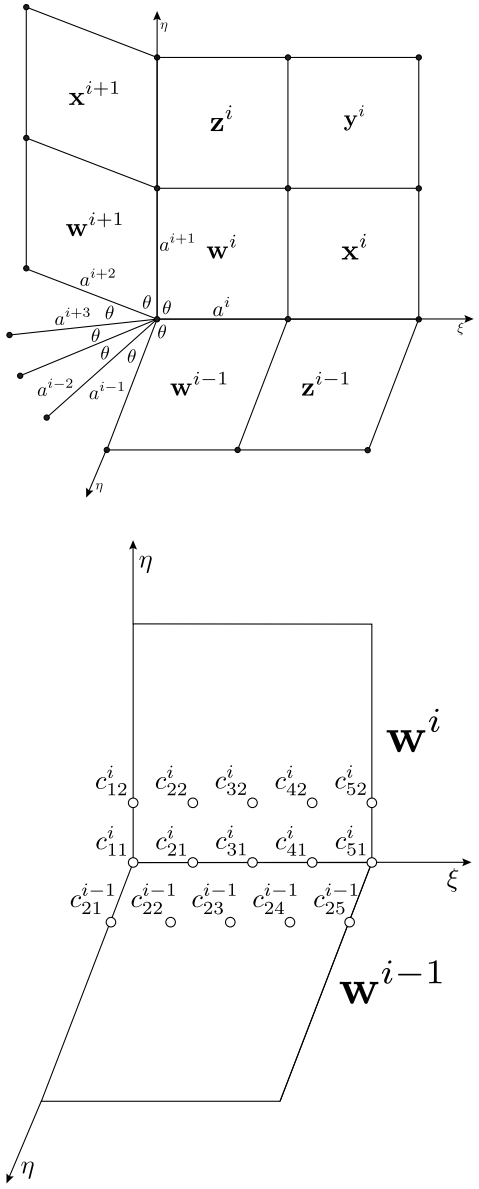


Figure 6.5: The i^{th} quadrant of a two-ring neighborhood around an extraordinary point (top) and a close-up of the extraction coefficients involved in the G^1 constraint equations (bottom). The $\mathbf{w}^i$, $\mathbf{x}^i$, $\mathbf{y}^i$, and $\mathbf{z}^i$ represent Bézier elements, the a^i represent knot intervals, and the $c^i_{\alpha\beta}$ denote extraction coefficients used in the G^1 constraint equations (6.18) – (6.23).

In addition to the C^1 constraints along the boundaries of the one-ring neighborhood we also enforce G^1 continuity along spoke edges. In other words, we must derive the constraint equations such that Bézier elements $\mathbf{w}^{i-1}$ and $\mathbf{w}^i$ are G^1-continuous for $i = 1, \ldots, N$. The general necessary and sufficient conditions for two Bézier elements to be G^1 is that they have the same normal vector along their common boundary curve. Bézier elements $\mathbf{w}^{i-1}$ and $\mathbf{w}^i$ in Figure 6.5 are G^1 if there exists functions $a^i(\xi)$, $b^i(\xi)$, and $c^i(\xi)$ that satisfy

$$f^i(\xi) = a^i(\xi)\frac{\partial \mathbf{w}^{i-1}(\xi,\eta)}{\partial \eta}|_{\eta=0} + b^i(\xi)\frac{\partial \mathbf{w}^{i}(\xi,\eta)}{\partial \xi}|_{\eta=0} + c^i(\xi)\frac{\partial \mathbf{w}^{i}(\xi,\eta)}{\partial \eta}|_{\eta=0} \equiv 0 \tag{6.11}$$

$$= a^i(\xi)\mathbf{w}_\eta^{i-1}(\xi) + b^i(\xi)\mathbf{w}_\xi^{i}(\xi) + c^i(\xi)\mathbf{w}_\eta^{i}(\xi) \equiv 0. \tag{6.12}$$

We assign a shared local coordinate system for both elements where the local ξ parameter lies along their common boundary curve as shown in Figure 6.5. Throughout, we adopt the notation

$$\langle c_1, c_2, \ldots, c_{p+1}\rangle^p(\xi) = \sum_{\alpha=1}^{p+1} c_\alpha B_\alpha^p(\xi) \tag{6.13}$$

where $B_\alpha^p(\xi)$ is a Bernstein polynomial of degree p in ξ. For simplicity, we will drop the superscript A. If $\mathbf{w}^{i-1}(\xi,\eta)$ and $\mathbf{w}^i(\xi,\eta)$ are biquartic then $a^i(\xi) = a^{i+1}$ and $c^i(\xi) = a^{i-1}$. In addition, $b^i(\xi)$ must be quadratic and the common boundary curve must be cubic. Then, the individual terms in (6.12) can be written as

$$a^i(\xi)\mathbf{w}_\eta^{i-1}(\xi) = 4a^{i+1}\langle c_{21}^{i-1} - c_{11}^i, c_{22}^{i-1} - c_{21}^i, c_{23}^{i-1} - c_{31}^i, c_{24}^{i-1} - c_{41}^i, c_{25}^{i-1} - c_{51}^i\rangle^4(\xi), \tag{6.14}$$

$$b^i(\xi)\mathbf{w}_\xi^{i}(\xi) = \langle \lambda^i, 0, 0\rangle^2(\xi)\langle 4(c_{21}^i - c_{11}^i), 4(\frac{1}{4}c_{11}^i - c_{21}^i + c_{41}^i - \frac{1}{4}c_{51}^i), 4(c_{51}^i - c_{41}^1)\rangle^2(\xi) \tag{6.15}$$

$$= \lambda^i\langle 4(c_{21}^i - c_{11}^i), (\frac{1}{2}c_{11}^i - 2c_{21}^i + 2c_{41}^i - \frac{1}{2}c_{51}^i), \frac{2}{3}(c_{51}^i - c_{41}^i), 0, 0\rangle^4(\xi), \tag{6.16}$$

and

$$c^i(\xi)\mathbf{w}^i_\eta(\xi) = 4a^{i-1}\langle c^i_{12} - c^i_{11}, c^i_{22} - c^i_{21}, c^i_{32} - c^i_{31}, c^i_{42} - c^i_{41}, c^i_{52} - c^i_{51}\rangle^4(\xi). \tag{6.17}$$

The equation $f^i(\xi)$ is a degree four polynomial in ξ which vanishes only if the following five terms vanish,

$$a^{i+1}(c^{i-1}_{21} - c^i_{11}) + \lambda^i(c^i_{21} - c^i_{11}) + a^{i-1}(c^i_{12} - c^i_{11}) = 0, \tag{6.18}$$

$$4a^{i+1}(c^{i-1}_{22} - c^i_{21}) + \lambda^i(\frac{1}{2}c^i_{11} - 2c^i_{21} + 2c^i_{41} - \frac{1}{2}a^i_{51}) + 4a^{i-1}(c^i_{22} - c^i_{21}) = 0, \tag{6.19}$$

$$4a^{i+1}(c^{i-1}_{23} - c^i_{31}) + \frac{2}{3}\lambda^i(c^i_{51} - c^i_{41}) + 4a^{i-1}(c^i_{32} - c^i_{31}) = 0, \tag{6.20}$$

$$a^{i+1}(c^{i-1}_{24} - c^i_{41}) + a^{i-1}(c^i_{42} - c^i_{41}) = 0, \tag{6.21}$$

$$a^{i+1}(c^{i-1}_{25} - c^i_{51}) + a^{i-1}(c^i_{52} - c^i_{51}) = 0. \tag{6.22}$$

To force the boundary curve to be a cubic polynomial requires that the fourth derivative vanishes. This constraint can be written as,

$$c^i_{11} - 4c^i_{21} + 6c^i_{31} - 4c^i_{41} + c^i_{51} = 0. \tag{6.23}$$

Notice that (6.18), (6.19), and (6.20) must be solved simultaneously for $i = 1, \ldots, N$. Equations (6.18), $i = 1, \ldots, N$, are often called *consistency conditions*. If we assume non-zero knot intervals, the solution for (6.18), $i = 1, \ldots, N$, requires that $c_{11}, c^1_{21}, \ldots, c^N_{21}$ all lie in a plane. Additionally, we impose the constraint that there exists an affine map that forces all angles marked θ in Figure 6.4

to be equal and $|c^i_{21} - c^i_{11}|$ to be proportional to the knot intervals a^i_0. Under those restrictions, (6.18) is satisfied if

$$\lambda^i = -2\frac{a^{i-1}a^{i+1}}{a^i}\cos\theta \tag{6.24}$$

where $\theta = \frac{2\pi}{N}$.

The global constraint equations (6.18) – (6.23) are assembled into $\mathbf{G}_A$ and $\mathbf{g}_A$ for $i = 1, \ldots, N$ using the local-to-global map (6.10).

6.2.2.2 Assembling the fairing system

In contrast to more complicated fairing functionals commonly used in CAGD [77] we employ a very simple and efficient fairing technique which minimizes the vertical and horizontal differences between the extraction coefficients resulting from generalized Bézier extraction and the perturbed extraction coefficients over each one-ring extraordinary element. In other words, the global equations

$$\tilde{c}^A_{I(i,\alpha,\beta)} - \tilde{c}^A_{I(i,\alpha+1,\beta)} = c^{i,A}_{\alpha\beta} - c^{i,A}_{\alpha+1\beta}, \quad 1 \leq \alpha \leq 4, 1 \leq \beta \leq 5 \tag{6.25}$$

and

$$\tilde{c}^A_{I(i,\alpha,\beta)} - \tilde{c}^A_{I(i,\alpha,\beta+1)} = c^{i,A}_{\alpha\beta} - c^{i,A}_{\alpha\beta+1}, \quad 1 \leq \alpha \leq 5, 1 \leq \beta \leq 4 \tag{6.26}$$

are assembled into $\mathbf{F}_A$ and $\mathbf{f}_A$ for $i = 1, \ldots, N$. Figure 6.6 shows the smoothed G^1-continuous T-spline basis function corresponding to the valence five extraordinary point in Figure 6.1. Notice the difference in smoothness when compared to the T-spline basis function resulting from generalized Bézier extraction as shown on the bottom of Figure 6.3.

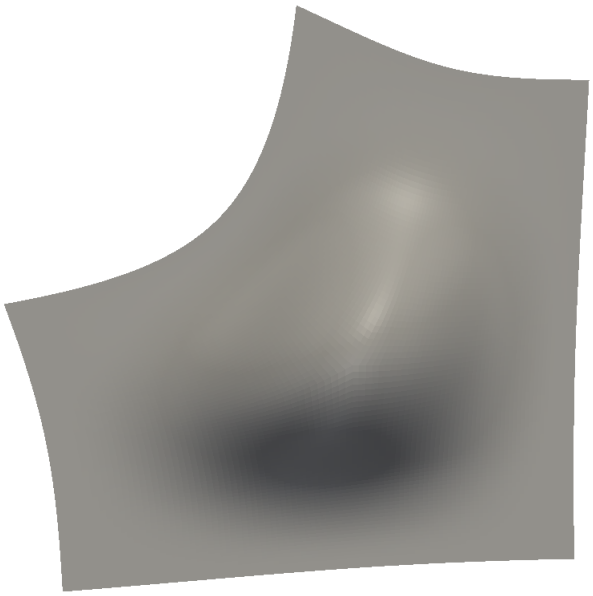


Figure 6.6: The smoothed G^1-continuous T-spline basis function corresponding to the valence five extraordinary point in Figure 6.1. Compare this result to the basis function on the bottom of Figure 6.3.

6.3 Analysis-suitability

Using our formulation for extraordinary points, it can be proven that a T-mesh with extraordinary points is analysis-suitable (i.e. the blending functions are linearly independent, form a partition of unity, etc.) if the regular T-mesh is analysis-suitable as described in Chapter 5 and all extraordinary points are separated by at least four T-mesh elements and no T-junction 1-bay face extension touches the interior of a T-mesh element in a three-ring neighborhood. As a result, each T-mesh element contains at most one extraordinary vertex. The T-mesh in Figure 6.1 is analysis-suitable.

6.4 Patch tests

To study of the behavior of T-splines near extraordinary points in an analysis setting we apply standard patch tests [7, 56]. Patch tests are used to determine whether an arbitrary "patch" of elements can exactly reproduce basic constant and linear deformation states. From a more practical standpoint, patch tests are also used to assess the correctness of a finite element implementation. Since *all* T-splines are affinely covariant and isoparametric by construction all patch tests should be satisfied exactly.

A simple patch test is illustrated in Figure 6.7 for a unit square with the various displacement boundary conditions listed on the right. The equations of isotropic linear elasticity are solved. A Poisson's ratio of 0.3 is assumed and Young's modulus, E, is one.

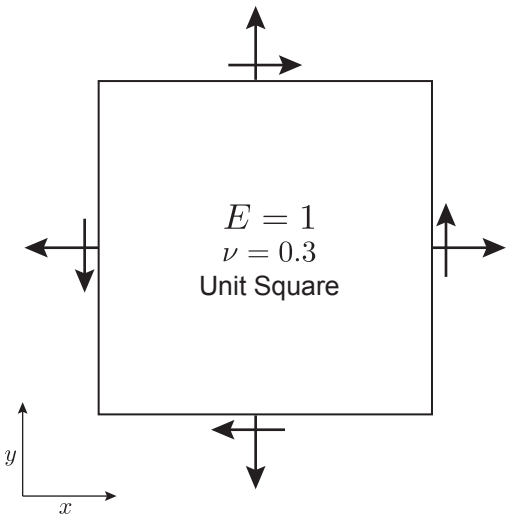
The T-mesh shown in Figure 6.8 is used in all cases. Initially, all knot intervals are one. Then, the knot intervals associated with the edges intersected by the dotted lines are increased while all other knot intervals remain the same. The

resulting Bézier meshes for $k = 1, 5, 10$ and 100 are shown in Figure 6.9. Notice the changing mesh spacing resulting from the increasing disparity between knot intervals.

The one-ring extraordinary elements are biquartic while all other Bézier elements are bicubic. To integrate over the Bézier elements we use Gaussian quadrature with a $p + 1$ rule where p is the polynomial degree of the *Bernstein* basis functions. Thus, in two-dimensions we use a 4-by-4 quadrature rule for bicubic Bernstein basis functions and a 5-by-5 quadrature rule for biquartic Bernstein basis functions. We note that more efficient quadrature schemes could probably be devised for the Bézier elements in the regular T-mesh [60] but this is not explored in this thesis.

The patch test results for $k = 1, 5$ and 10 are shown in Figures 6.10 – 6.13. We note that the Bézier mesh for $k = 1$ is plotted with the solution profile. In all cases, the patch tests are satisfied to machine precision. For the rigid translation and rotation patch tests shown in Figures 6.10 and 6.11 constant displacement profiles in the x and y directions are reproduced exactly and all stress states are zero. For the stretch patch tests shown in Figure 6.12 linear displacement profiles in the x and y direction are reproduced exactly and σ_{xx} and σ_{yy} are constant while all other stress states are zero. Finally, for the shear patch tests shown in Figure 6.13 linear displacement profiles in the x and y direction are reproduced exactly and σ_{xy} is constant while all other stress states are zero.

Interestingly, for $k = 100$, the Bézier elements near the valence five extraordinary point begin to invert (see Figure 6.9). The resulting negative Jacobians prevent us from using this mesh in analysis. This behavior has been observed generally when widely varying knot intervals are adjacent to an extraordinary point. As a rule of thumb, to prevent this behavior, the ratio of the largest to smallest knot in-



Rigid Translation in x case:
$u_x = 0.1$
$u_y = 0$
Rigid Translation in y case:
$u_x = 0$
$u_y = 0.1$
Rigid rotation case:
$u_x = 0.1y$
$u_y = -0.1x$
Stretch in x case:
$u_x = 0.1x$
$t_y = 0$
Stretch in y case:
$t_x = 0$
$u_y = 0.1y$
Horizontal shear case:
$u_x = 0.1y$
$u_y = 0$
Vertical shear case:
$u_x = 0$
$u_y = 0.1x$

Figure 6.7: Standard patch tests. All patch tests are performed on a unit square with a Poisson's ratio of 0.3 and a Young's modulus, E, of one. The box on the right specifies the boundary conditions for rigid translation, rigid rotation, stretch, and shear patch tests.

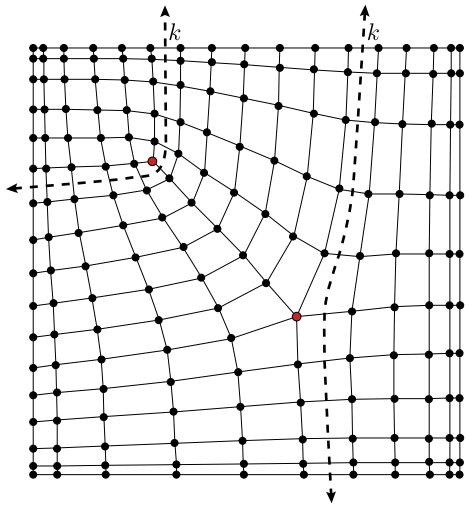


Figure 6.8: The T-mesh used for the patch tests in Figure 6.7. The knot intervals associated with the edges intersected by the dotted lines are varied resulting in the Bézier meshes in Figure 6.9.

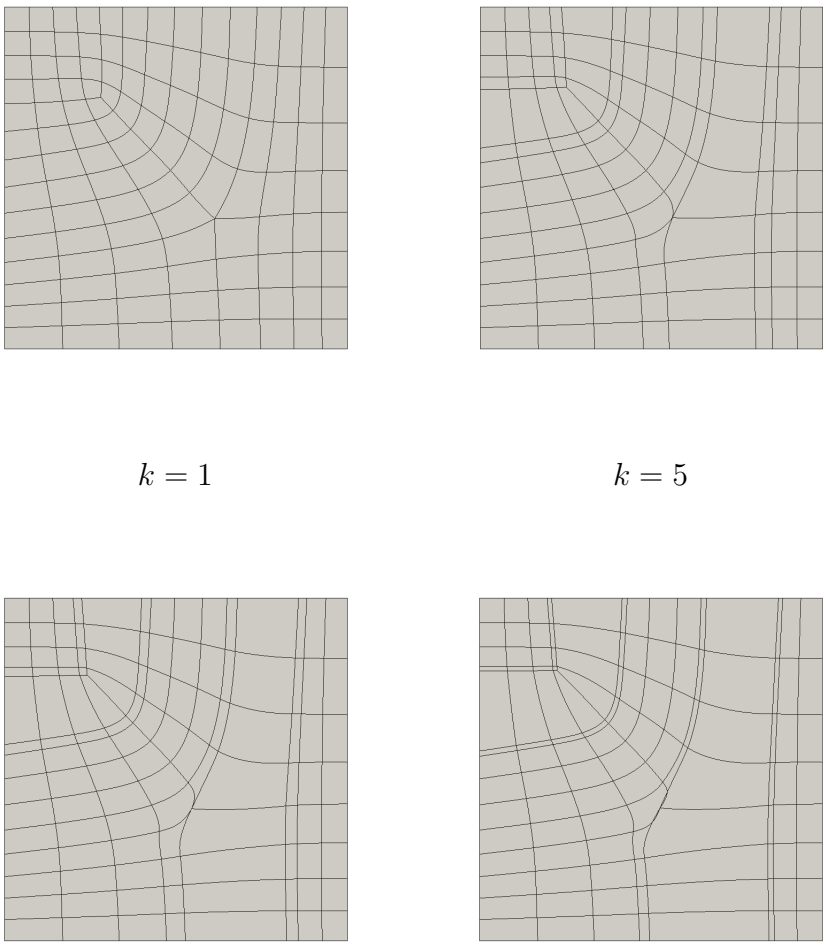
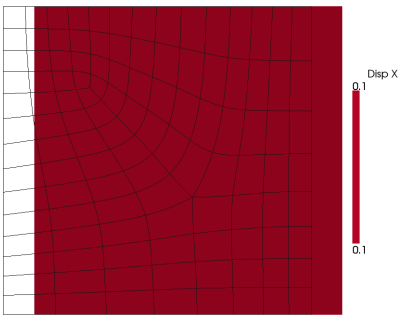
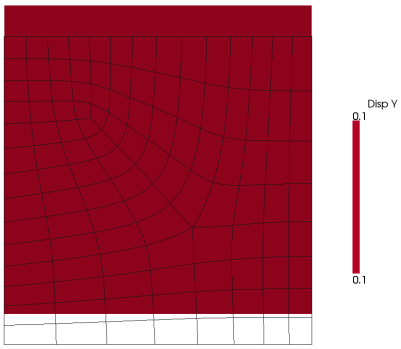


Figure 6.9: The Bézier meshes, corresponding to the T-mesh in Figure 6.8, for $k = 1, 5, 10$ and 100.



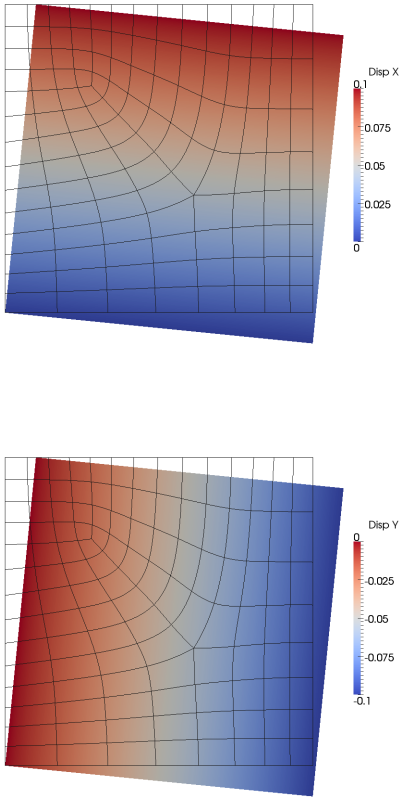
Rigid translation in x



Rigid translation in y

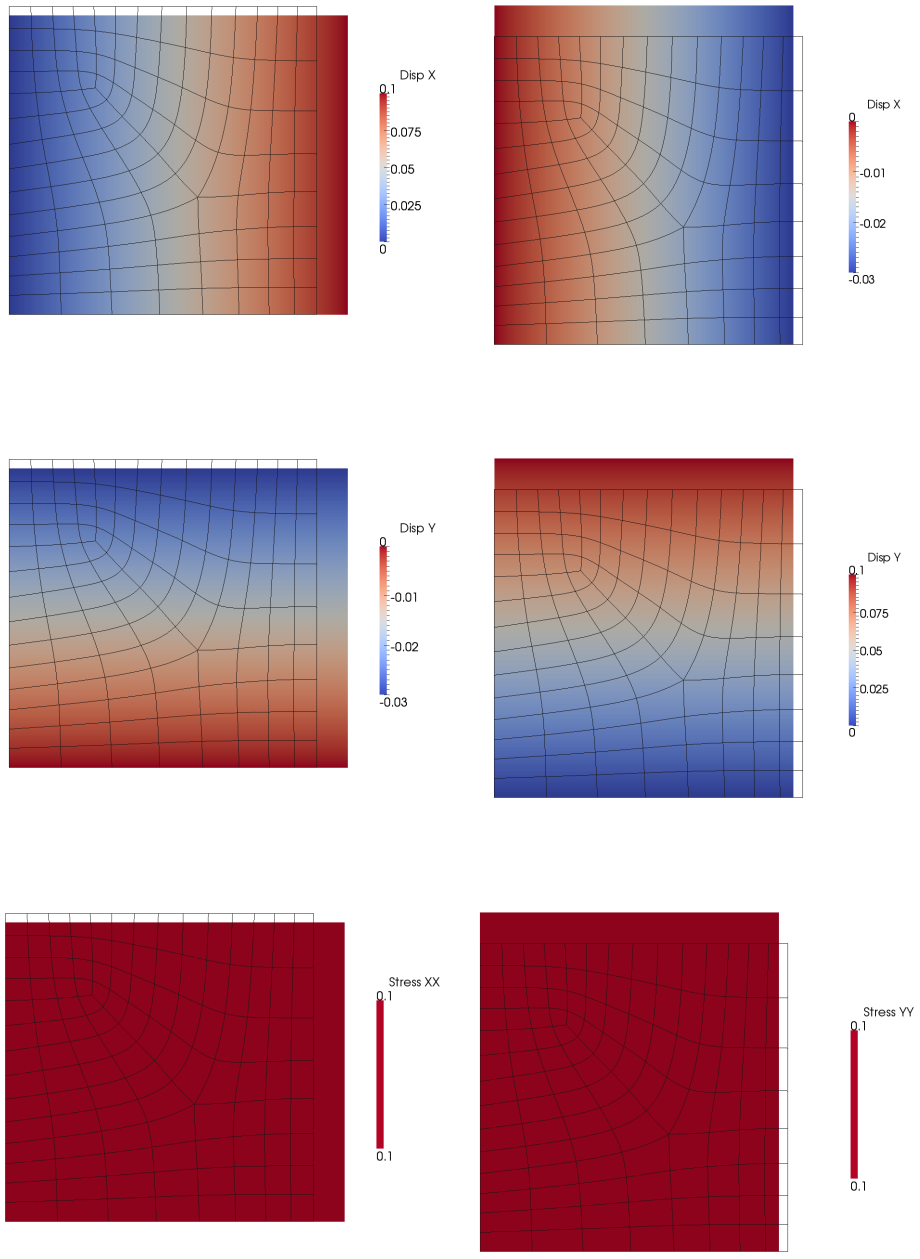
Figure 6.10: Results for the rigid translation patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Constant displacement profiles in the x and y direction are reproduced exactly and all stress states are zero.

terval corresponding to the spoke edges surrounding an extraordinary point should not be more than 5 to 1.



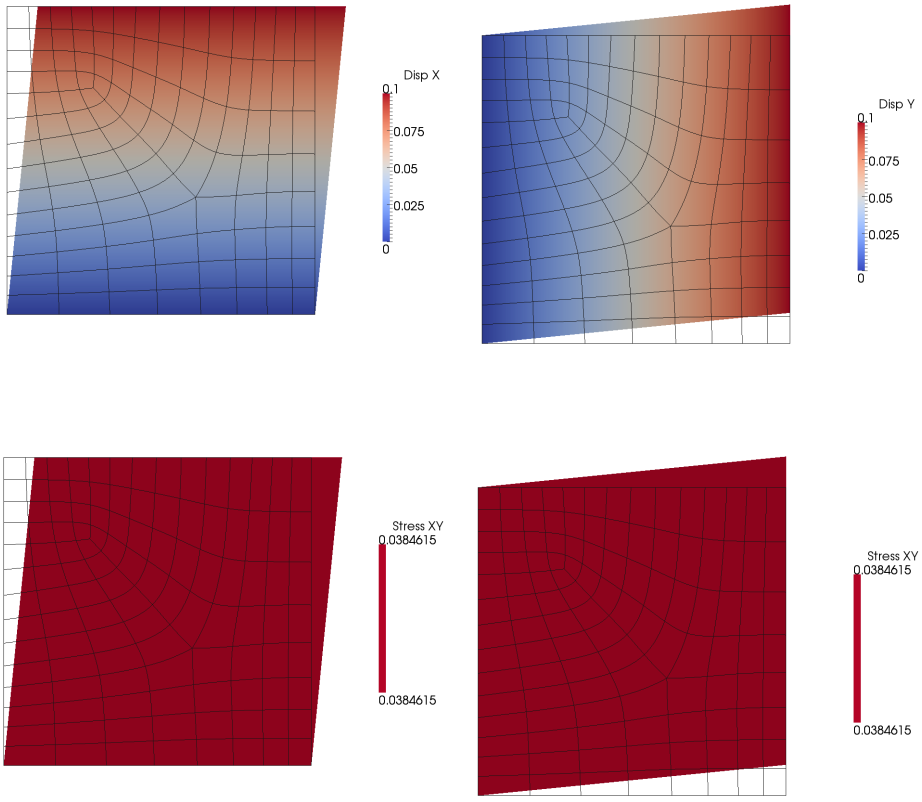
Rigid rotation

Figure 6.11: Results for the rigid rotation patch test. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y directions are reproduced exactly and all stress states are zero.



Stretch in x Stretch in y

Figure 6.12: Results for the stretch patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y direction are reproduced exactly and σ_{xx} and σ_{yy} are constant while all other stress states are zero.



Shear in x Shear in y

Figure 6.13: Results for the shear patch tests in the x and y directions. This patch test is satisfied exactly using the T-spline in Figure 6.1. Linear displacement profiles in the x and y direction are reproduced exactly and σ_{xy} is constant while all other stress states are zero.

Chapter 7

T-spline-based isogeometric analysis

We now apply T-splines and isogeometric analysis to several applications in solids and fluids. We have chosen examples which demonstrate the various important features of T-spline discretizations. We start with simple benchmark problems in Sections 7.1 and 7.2 which facilitate direct comparison with NURBS. These examples demonstrate that T-splines possess similar convergence properties to NURBS with far fewer degrees of freedom. Additionally, these examples employ T-spline surfaces and *solids* of *arbitrary* degree.

We then apply T-splines to a continuum damage model in Section 7.3. We study the effect of higher-order smoothness on accuracy and robustness of gradient damage formulations. We use analysis-suitable local refinement to resolve the damage zone while preserving smoothness and exact geometry.

Finally, we develop an adaptive isogeometric analysis framework which couples analysis-suitable T-splines, local refinement, and Bézier extraction. We demonstrate the potential of the approach by applying it to a dynamic phase-field fracture model. To capture the behavior of cracks accurately, robust and efficient element technology and highly localized refinement algorithms are essential. These examples also allow us to study the scalability of T-spline element technology to very large problems in two and three dimensions and parallel implementations.

7.1 Isogeometric fluids analysis

Isogeometric analysis of fluid flows with NURBS discretizations is a well-studied topic, with applications in turbulence modeling [2,10], cardiovascular flows [12, 121], and fluid-structure interaction [11, 12, 14]. It has been shown in these works that NURBS functions exhibiting higher-order continuity are an ideal candidate for approximating such flows.

The model problem which we consider here is the linear advection-reaction-diffusion equation:

$$cu + \mathbf{a} \cdot \nabla u - \nabla \cdot (\kappa \nabla u) = f \tag{7.1}$$

where u denotes the concentration, c is the reaction coefficient, $\mathbf{a}$ is the velocity, κ is the diffusivity, and f is the source.

We consider two opposite regimes of the advection-reaction-diffusion system:

- reaction-diffusion ($\mathbf{a} = 0$), and
- advection-diffusion ($c = 0$).

The first example involves the simulation of the reaction-diffusion problem described in Figure 7.1. We note that the boundary condition is zero except near the corners. Since this particular problem is strongly dominated by reaction, we expect that the solution is zero except near the corners, where it rapidly spikes to one. With local refinement, we hope to resolve these spikes near the corners.

The meshes we utilized for solving this problem are presented in Figure 7.2, and numerical results were obtained using Galerkin's method. Solution profiles for polynomial orders $p = 1, 2, 3$ are presented in Figures 7.3 - 7.5. With increasing

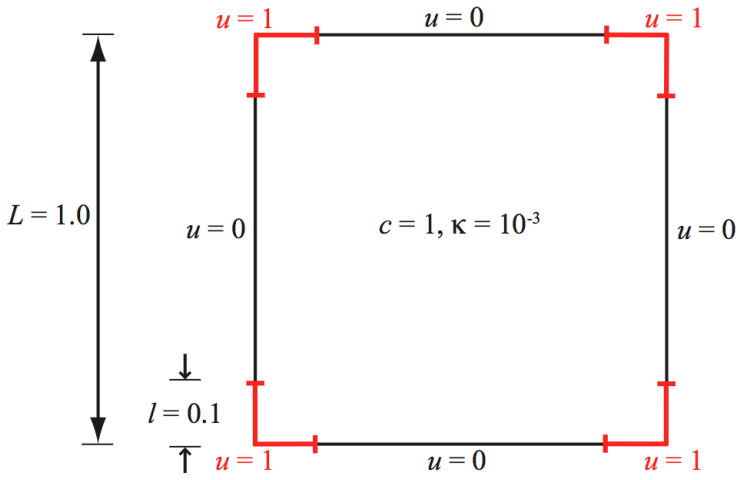


Figure 7.1: Reaction diffusion problem. Problem description and data.

polynomial order and local refinement, we see that not only are we able to more sharply capture the corner phenomena but we are also able to eliminate spurious oscillations.

The second example involves solution of the advection-diffusion problem shown in Figure 7.6. Here, the Péclet number, which characterizes competition between advection and diffusion, is 10^{-6}, making the problem strongly advection-dominated, and thus we expect a sharp interior layer and sharp boundary layers at the outflow. Successful capturing of such layers requires robust, stable numerical techniques in addition to increased resolution. The problem was investigated using NURBS-based isogeometric analysis, SUPG stabilization [26], and k-refinement in [57]. There, it was noted that globally k-refined meshes produced results that were nearly monotone. Here, we investigate the effect of local h-refinement on the solution.

This is also an ideal problem to demonstrate the behavior of analysis-suitable

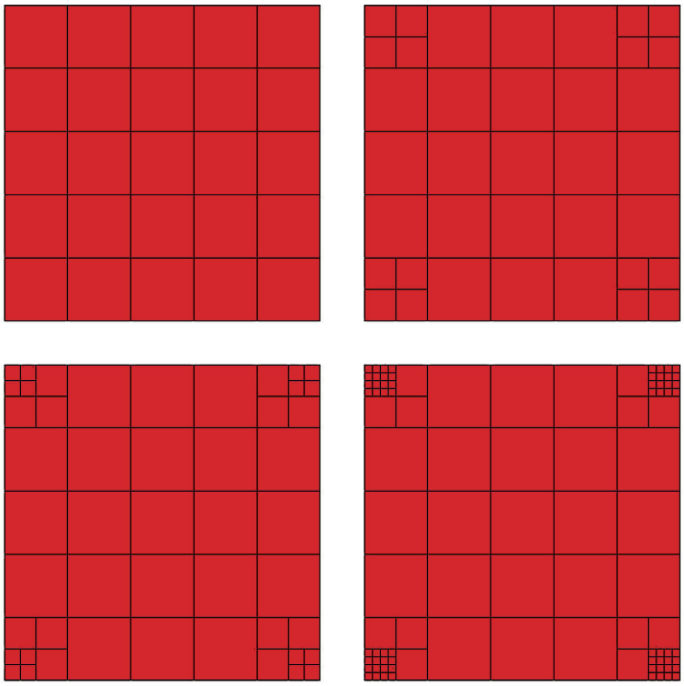


Figure 7.2: Reaction diffusion problem. Sequence of T-spline meshes.

local refinement. This is due to the anisotropic orientation of the interior layer with respect to the linear parameterization of the geometry. It was noted in [9, 44] that employing the refinement scheme introduced in [104] for this problem (for $p > 1$) generated many superfluous control points.

We begin with a cubic uniform mesh of 8 x 8 Bézier elements. We then employ an automatic refinement scheme that makes use of a simple gradient-based error indicator and the adaptive scheme described in Section 5.8. In all cases, the standard SUPG formulation is used with stabilization parameter $\tau = h_a/2a$, where h_a is the integration element length in the direction of the flow velocity. For the problem considered, $a = |\mathbf{a}| = 1$ and $h_a = \sqrt{2}h$, where h is the element edge length.

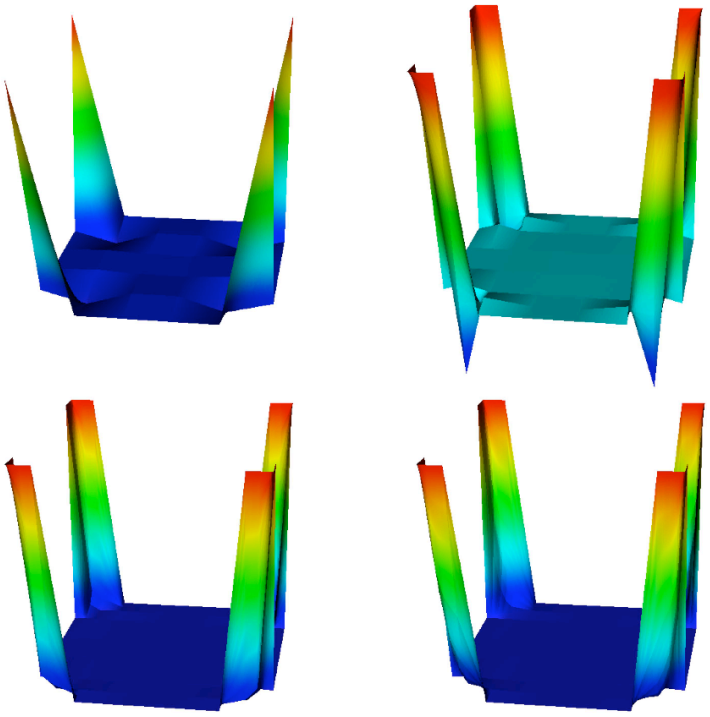


Figure 7.3: Reaction diffusion problem. Results for $p = 1$.

The Bézier meshes and corresponding solutions for each refinement step are shown in Figure 7.7. We find that using T-splines and local refinement the layers become sharper, and the overshoots and undershoots about the layers are attenuated as expected. For this problem, we also ran the refinement algorithm from [104] for comparison. The results are shown in Table 7.1 where we denote the results using the algorithm from [104] by "old" and the results using the analysis-suitable local refinement algorithm by "new." The new algorithm outperforms the old algorithm in terms of Bézier elements and basis functions introduced through refinement.

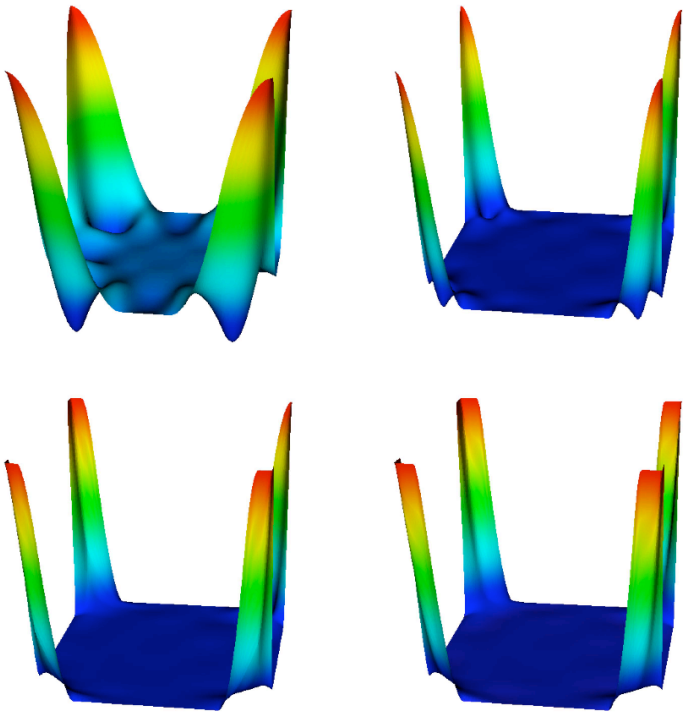


Figure 7.4: Reaction diffusion problem. Results for $p = 2$.

	Funcs		Elems	
Run	Old	New	Old	New
1	334	297	256	222
2	1090	704	1017	637
3	3749	1721	3757	1744

Table 7.1: A comparison of the "old" local refinement algorithm from [104] and the "new" analysis-suitable local refinement algorithm in terms of Bézier elements and basis functions introduced through refinement.

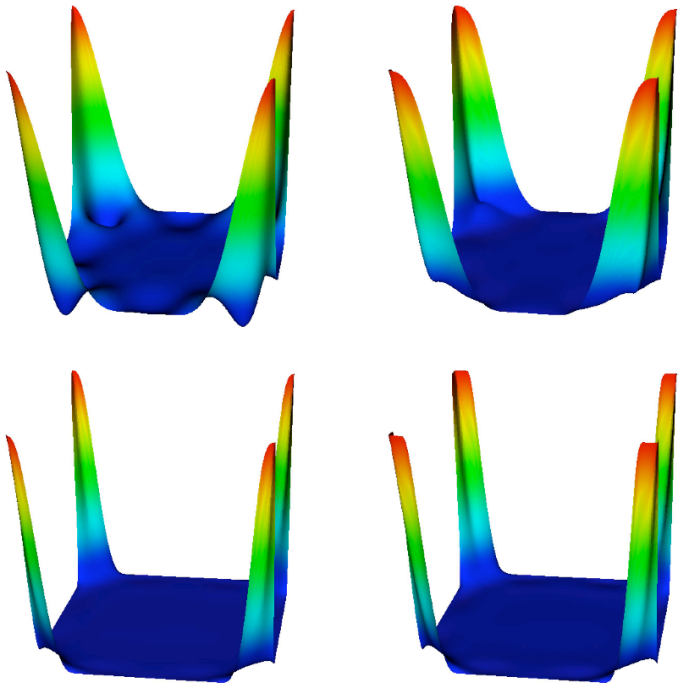


Figure 7.5: Reaction diffusion problem. Results for $p = 3$.

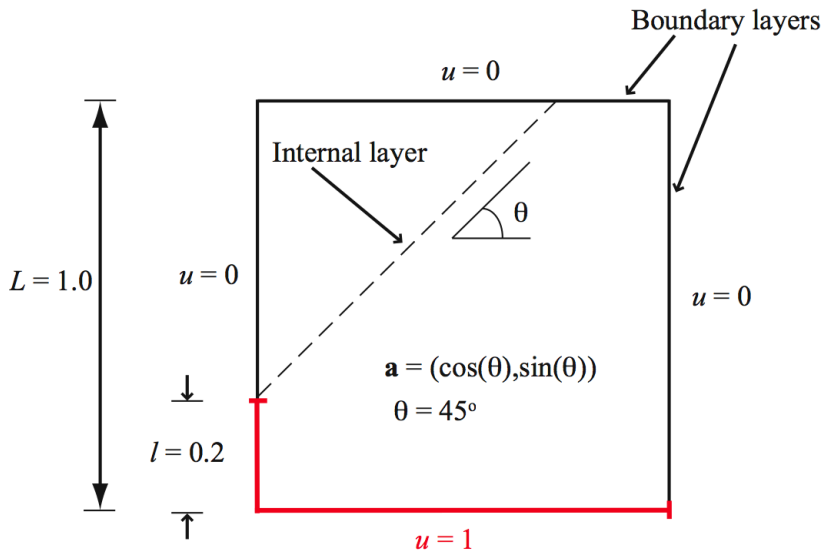


Figure 7.6: Advection diffusion problem, $\theta = 45°$. Problem description and data.

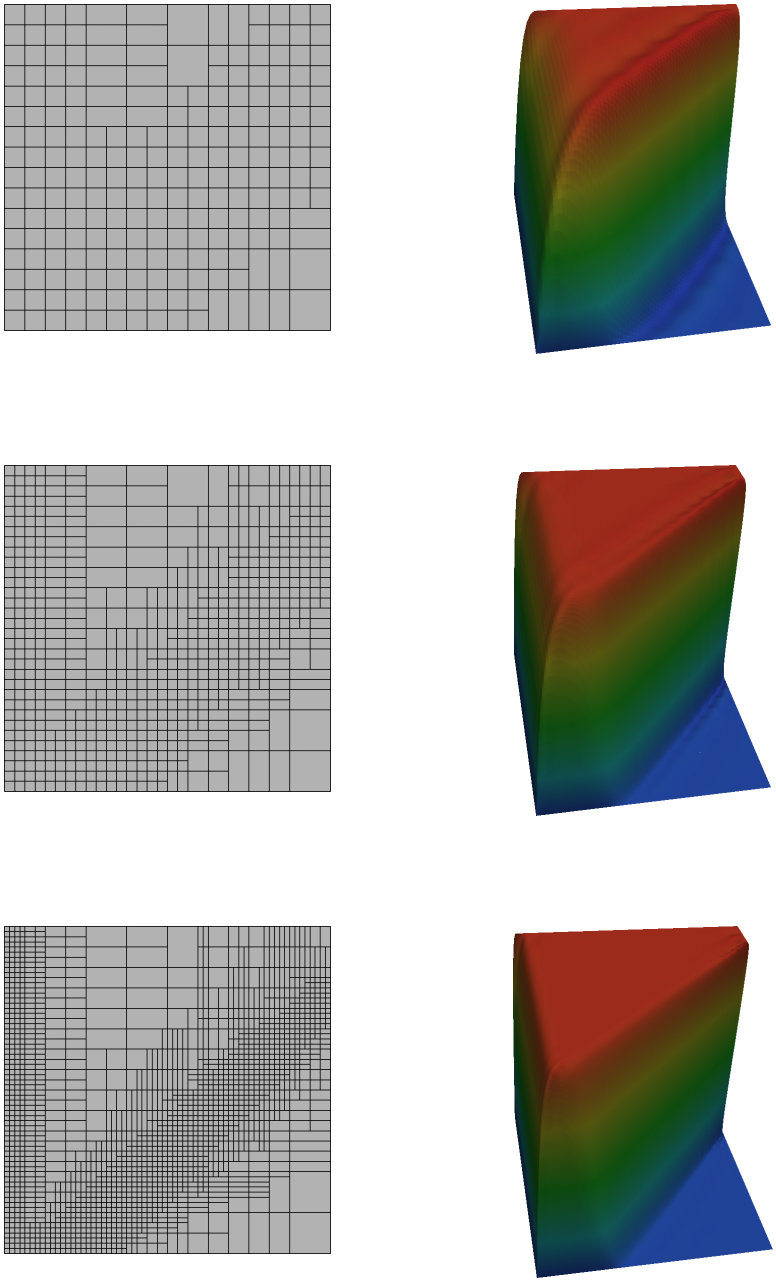


Figure 7.7: Advection skew to the mesh, $\theta = 45°$. Results for $p = 3$. The Bézier meshes are on the left and the corresponding solutions are on the right.

7.2 Isogeometric structural analysis

An isogeometric structural analysis framework based on T-splines preserves many of the desirable properties of its NURBS counterpart. In particular, T-splines also satisfy standard patch tests. In what follows, we present numerical solutions for linear elastic solids and structures. The Galerkin formulation of linear elasticity is employed. All the calculations involve thin shells, but these are modeled as three-dimensional solids and no shell assumptions are employed. A direct algebraic equation solver was employed for each of the calculations.

The first two examples come from the so-called shell obstacle course: the pinched hemisphere and the pinched cylinder. These problems, and their relevance to the assessment of shell analysis procedures, have been discussed extensively in the literature, and the particular form of the problems we have chosen is adapted from Felippa [49] and Belytschko et al. [17]. These problems were chosen due to the local nature of their external forcing (in fact, in both examples point loads are applied). The last example we consider is the hemispherical shell with stiffener presented in Rank et al. [91] who solved the problem using a finite element method and p-refinement strategy. In each of these examples, rather than using an automatic refinement strategy, we hand-crafted a sequence of T-meshes for various polynomial orders.

7.2.1 Pinched hemisphere

In the pinched hemisphere, equal and opposite concentrated point load forces are applied at antipodal points of the equator. The equator is otherwise considered to be free (see Figure 7.8). An example sequence of T-spline meshes is shown in Figure 7.9. Due to symmetry, only one quadrant is meshed. In this case, two quadratic T-spline elements were employed in the through-thickness direction (see Hughes,

Cottrell, and Bazilevs [57]). Quadratic through quintic T-spline surfaces were employed. A contour plot of the displacement on the deformed configuration is shown in Figure 7.10. Convergence of displacement under the inward directed load for the various orders is presented in Figure 7.11. As in the case of NURBS [57], the quadratic case converges very slowly and higher order functions lead to much faster convergence to the exact solution.

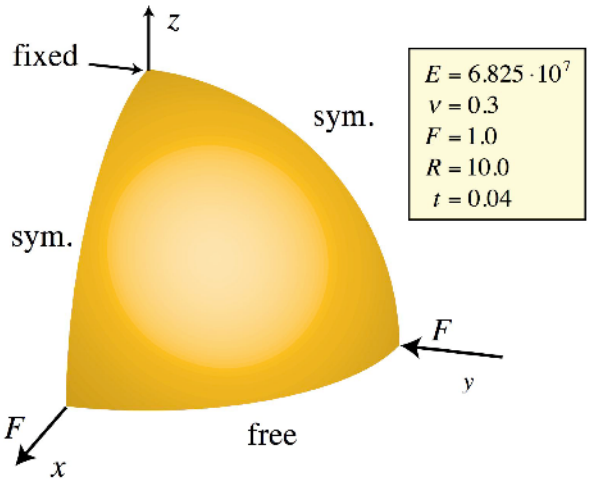


Figure 7.8: Shell Obstacle Course. Pinched hemisphere problem description and data.

7.2.2 Pinched cylinder

The pinched cylinder is subjected to equal and opposite concentrated forces at its midspan (see Figure 7.12). The ends are supported by rigid diaphragms. This constraint results in highly localized deformation under the loads. Only one octant of the cylinder is used in the calculations due to symmetry. As in the case of the pinched hemisphere, two quadratic Bézier elements were utilized in the through-thickness direction and quadratic through quintic orders of T-splines were utilized

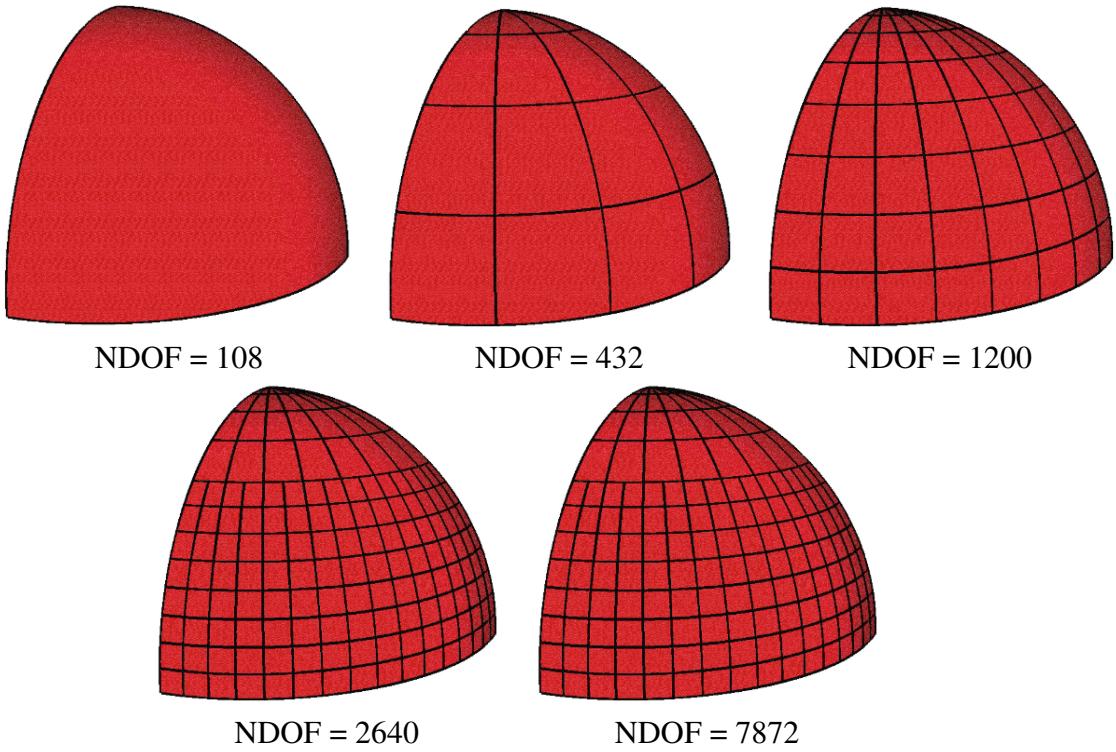
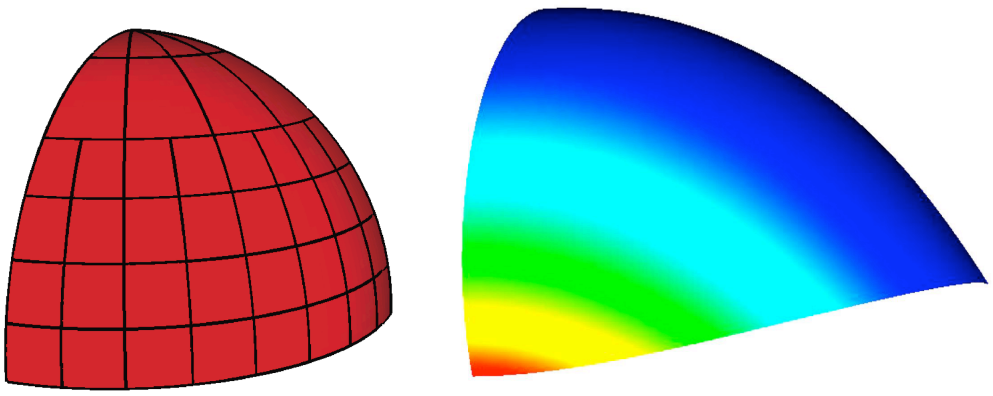


Figure 7.9: Shell Obstacle Course. Sequence of T-spline meshes for pinched hemisphere for $p = 2$.

on the surface. An example sequence of T-spline meshes is shown in Figure 7.13. A contour plot of the displacement on the deformed configuration is shown in Figure 7.14. Convergence of the displacement under the load is plotted in Figure 7.15. It is well known that, as long as the characteristic surface element dimension is large compared with the thickness, formulations which permit transverse shear deformations typically closely approximate formulations which satisfy the Kirchhoff constraint (i.e., zero transverse shear) [58].



(a) (b)

Figure 7.10: Shell Obstacle Course. Pinched hemisphere with $p = 5$, NDOF = 1524: (a) Physical mesh and (b) Displacement contours in direction of inward directed point load on deformed configuration (scaling factor of 30 used).

7.2.3 Hemispherical shell with a stiffener

The hemispherical shell with stiffener is subjected to gravity loading and external pressure, with the bottom surface fixed in the vertical direction (see Figure 7.16). Only a quarter of the domain is modeled due to symmetry. The initial mesh is constructed using rational quadratic T-splines in the circumferential direction and cubic T-splines in the other two directions and is shown in Figure 7.17. A series of refined meshes is shown in Figure 7.18. We found that no refinement was needed in the circumferential direction due to the axisymmetry of the solution and the fact that an exact geometry is employed. Figure 7.20 shows the vertical displacement and von Mises stress on the deformed configuration.

The Euclidean norm of the displacement and the von Mises stress were calculated at points A-D, identified in the problem description (Figure 7.16). Results for sequences of T-spline meshes are plotted in Figure 7.21 and 7.22 along with results from Rank et al. [91] for their finest simulation ($p = 8$) which we take as a

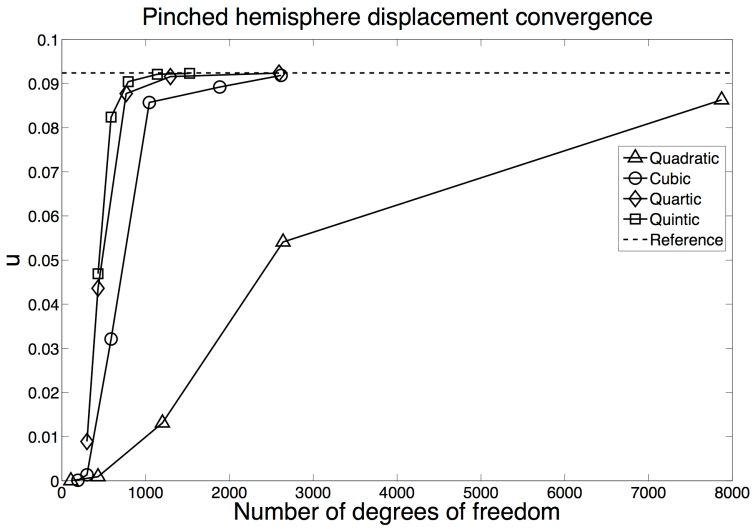


Figure 7.11: Shell Obstacle Course. Pinched hemisphere displacement convergence.

reference. Good agreement in the converged displacements and von Mises stresses is observed for cases except for the von Mises stress at point A, in which case the present result of the finest mesh employed is somewhat higher than the result of Rank et al. [91].

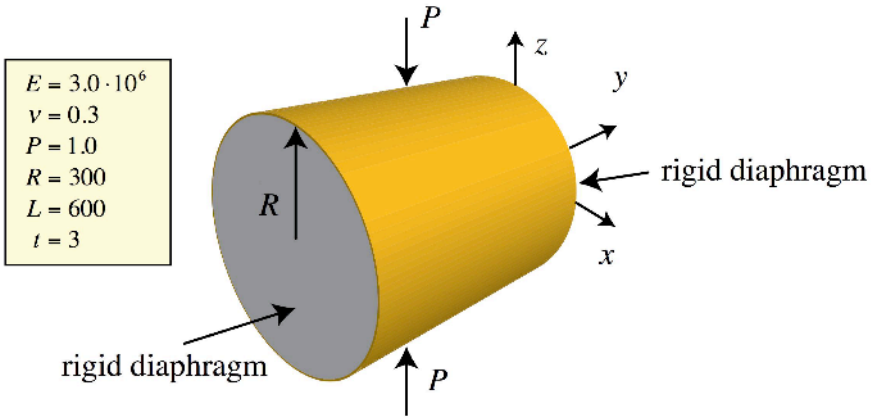


Figure 7.12: Shell Obstacle Course. Pinched cylinder problem description and data.

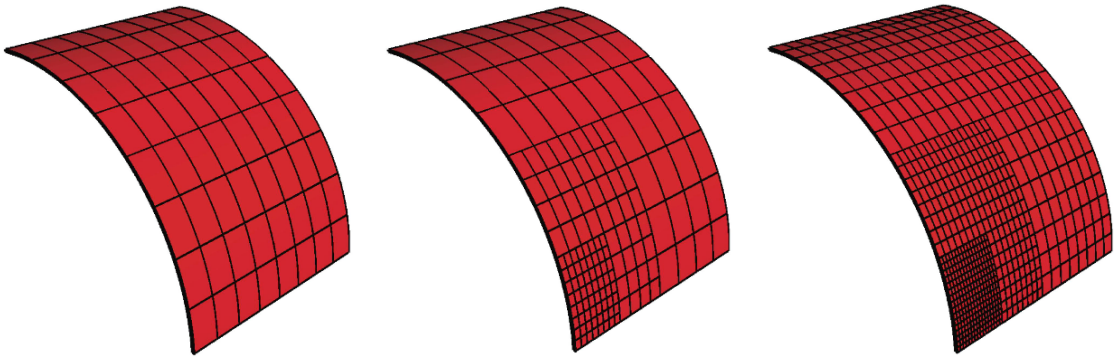
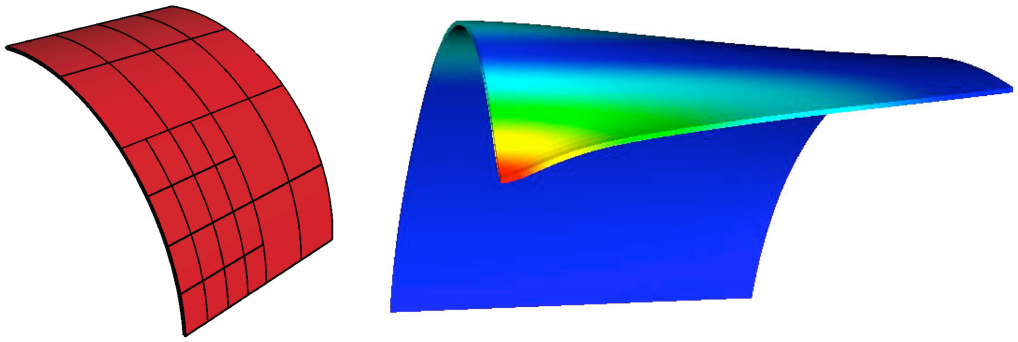


Figure 7.13: Shell Obstacle Course. Sequence of T-spline meshes for pinched cylinder for $p = 2$.



(a) (b)

Figure 7.14: Shell Obstacle Course. Pinched cylinder with $p = 5$, NDOF = 1260: (a) Physical mesh and (b) Displacement contours on deformed configuration (scaling factor of 3×10^6 used).

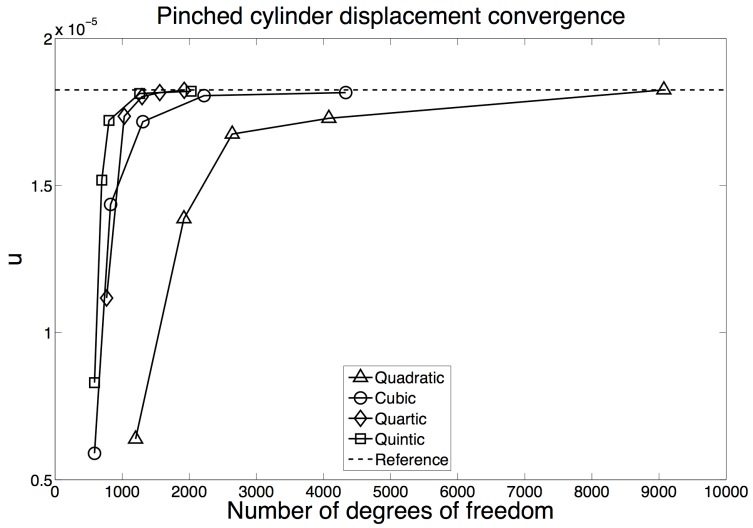
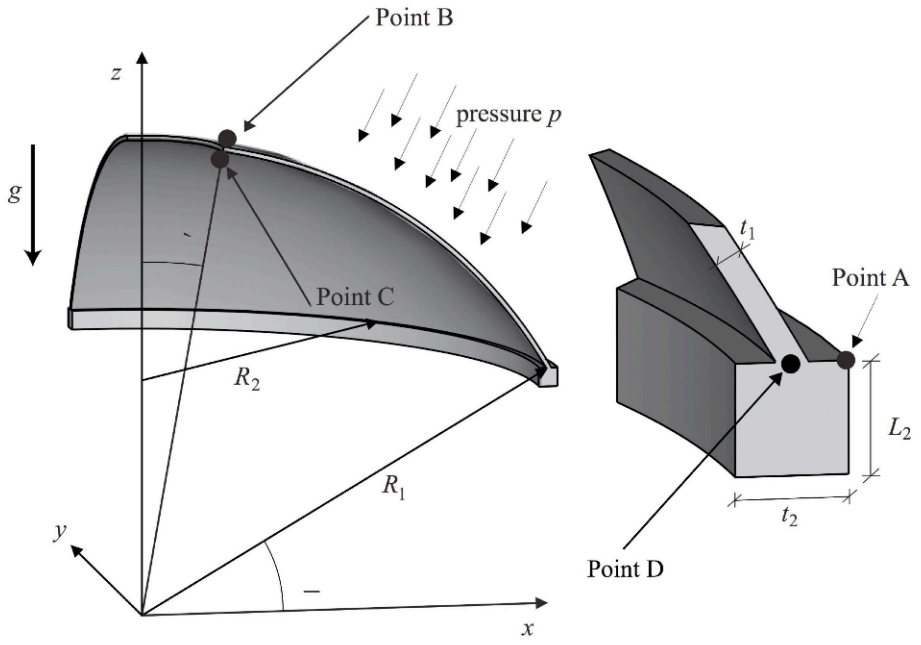


Figure 7.15: Shell Obstacle Course. Pinched cylinder displacement convergence.



Boundary conditions:

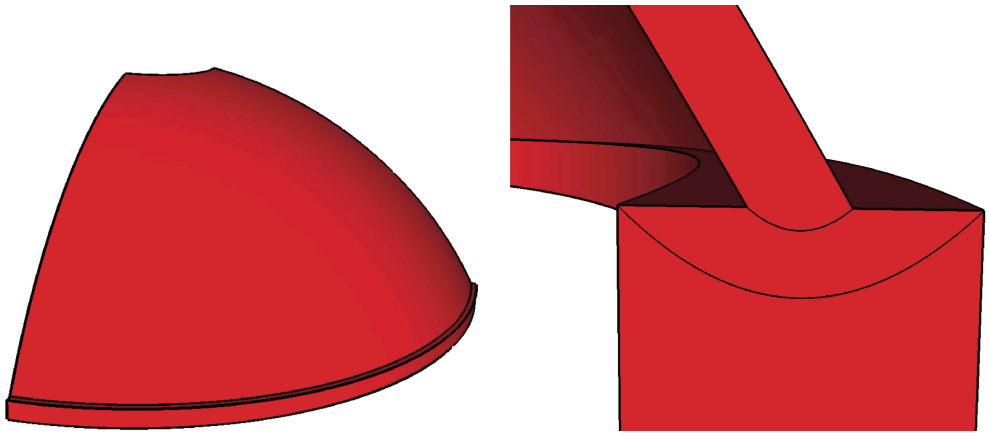
At bottom surface of stiffener:
$u_z = 0$

Symmetry at x-z-plane:
$u_y = 0$

Symmetry at y-z-plane:
$u_x = 0$

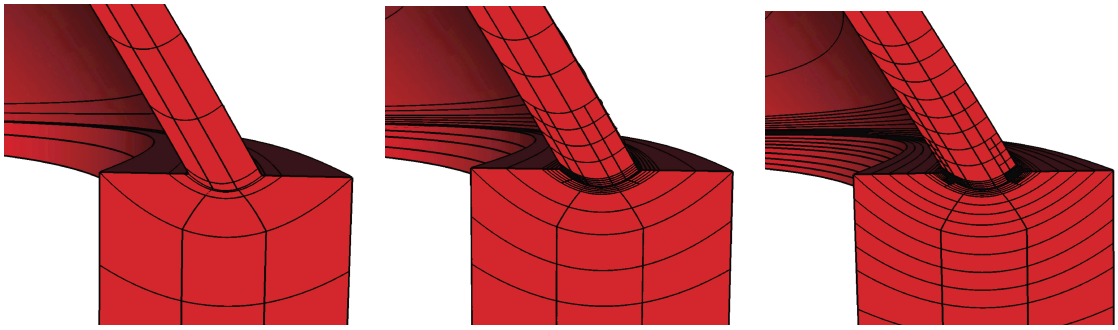
α	=	$30°$
β	=	$10°$
R_1	=	$10m$
R_2	=	$5\sqrt{3}m$
t_1	=	$0.1m$
t_2	=	$0.4m$
L_2	=	$0.4m$
E	=	$6.825 \cdot 10^7 \frac{kN}{m^2}$
ν	=	0.3
ρ	=	$500 \frac{kg}{m^3}$
g	=	$10.0 \frac{m}{s^2}$
p	=	$100.0 \frac{kN}{m^2}$

Figure 7.16: Hemispherical shell with stiffener. Problem description from Rank et al. [91].



(a) (b)

Figure 7.17: Hemispherical shell with stiffener. Initial mesh, NDOF = 360: (a) Coarse mesh and (b) Detail of stiffener.



NDOF = 972 NDOF = 2340 NDOF = 4248

Figure 7.18: Hemispherical shell with stiffener. Refined meshes.

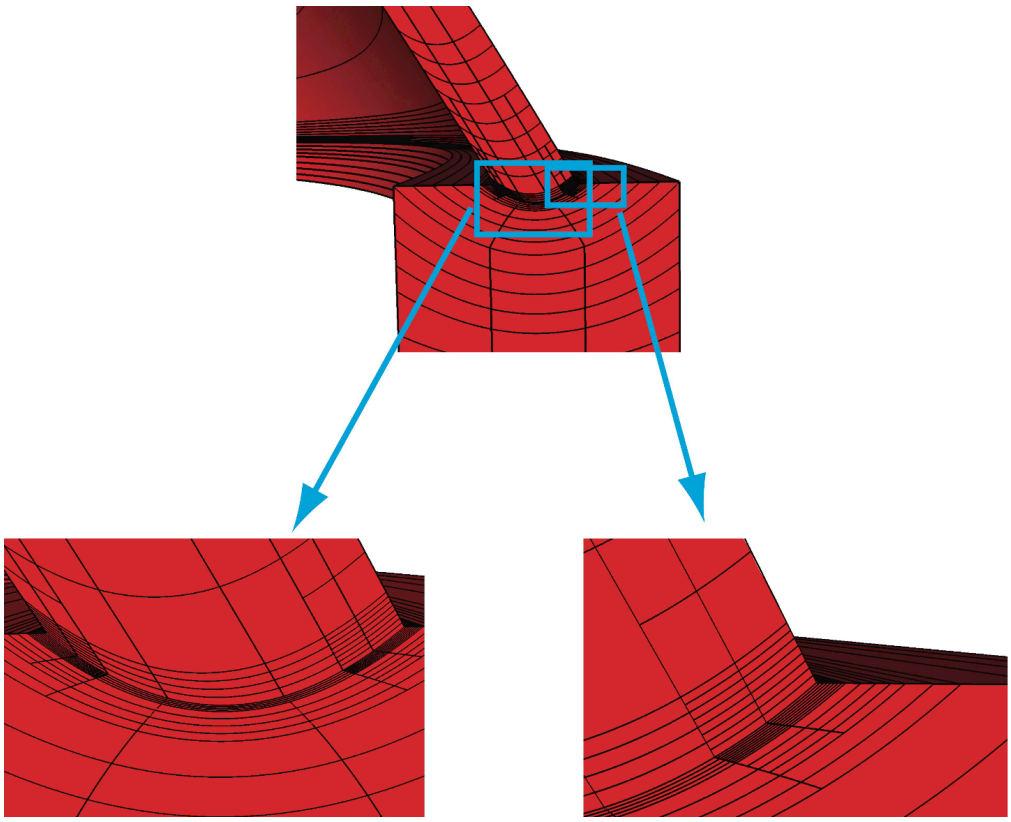
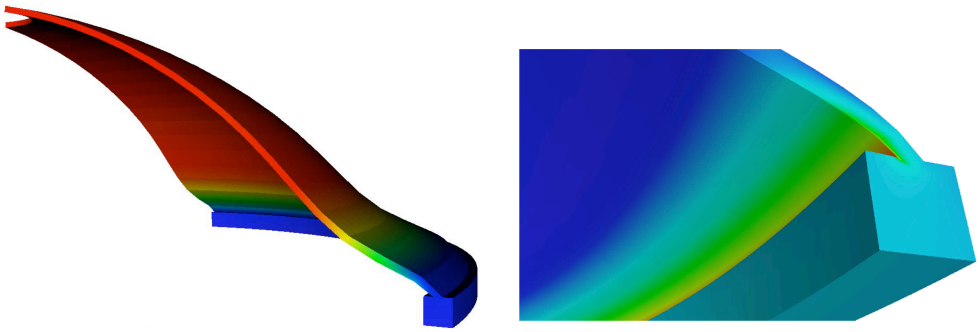
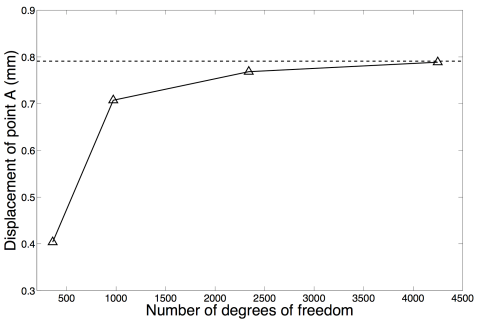


Figure 7.19: Hemispherical shell with stiffener. Detail of refinement for finest mesh (NDOF = 4248).

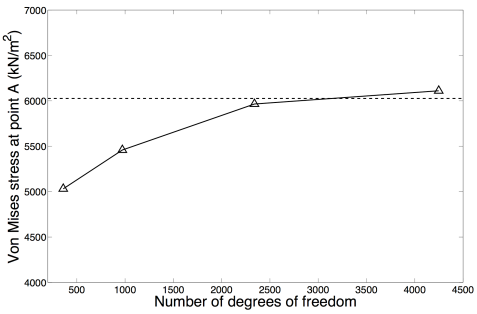


(a) (b)

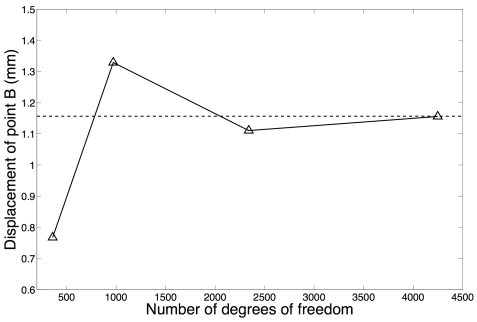
Figure 7.20: Finest mesh (NDOF = 4248): (a) Vertical displacement contours (scaling factor of 500 used) and (b) von Mises stress contours, detail of stiffener.



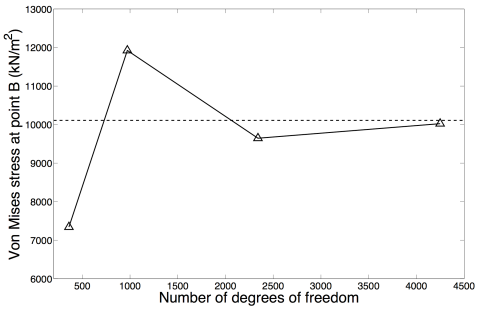
Point A Displacement



Point A Von Mises Stress

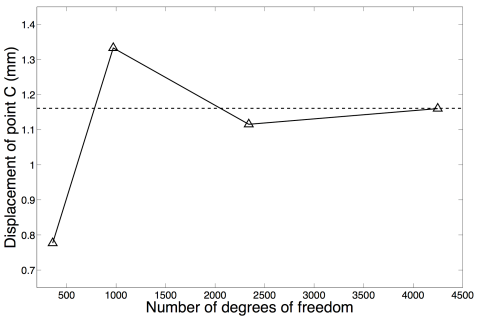


Point B Displacement

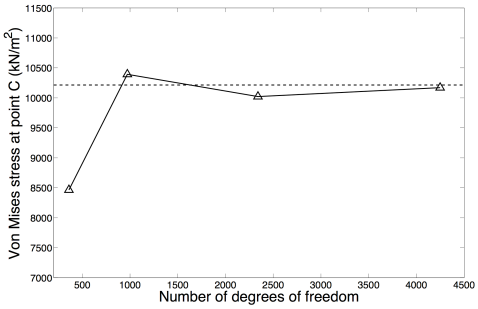


Point B Von Mises Stress

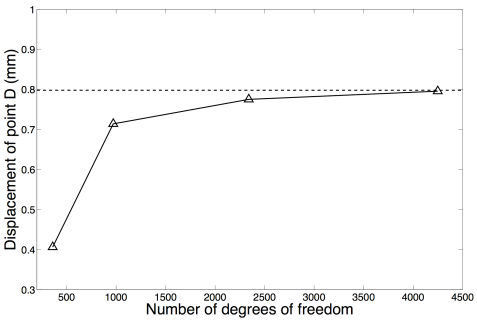
Figure 7.21: Hemispherical shell with stiffener. Convergence of displacement and von Mises stress at various points to benchmark solution [91].



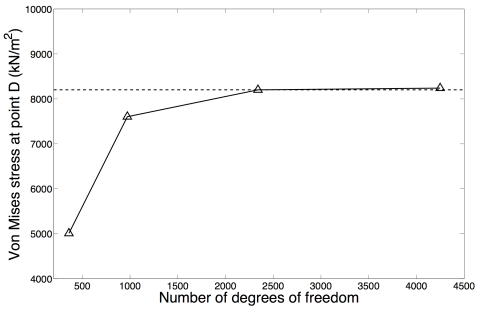
Point C Displacement



Point C Von Mises Stress



Point D Displacement



Point D Von Mises Stress

Figure 7.22: Hemispherical shell with stiffener. Convergence of displacement and von Mises stress at various points to benchmark solution [91].

7.3 Isogeometric gradient damage

Continuum damage models [68] are widely used for the simulation of diffuse fracture processes. Among the many variants, gradient damage formulations are perhaps the most popular. Among the gradient damage formulations, the implicit gradient enhancement [86] is considered the most effective. In its original form a second-order Taylor expansion is used to approximate a nonlocal smoothing integral, which results in a system of two second-order partial differential equations. This formulation is attractive from a discretization point of view since it can be solved using C^0-continuous finite elements. It has, however, been demonstrated that the accuracy of the second-order approximation can be limited [3, 55].

We use T-splines to study the effect of higher-order terms in the Taylor approximation of the nonlocal formulation, which result in higher-order gradient damage formulations. Specifically, we discretize the the second-order, fourth-order and sixth-order gradient formulations using T-splines. For additional details see [117].

7.3.1 Isotropic damage formulation

We consider a body $\Omega \subset \mathbb{R}^{d_s}$ with $d_s \in \{1, 2, 3\}$ and boundary $\partial\Omega$ (see Figure 7.31). The displacement of a material point $x \in \Omega$ is denoted by $u(x) \in \mathbb{R}^{d_s}$. The displacements satisfy Dirichlet boundary conditions, $u_i = \tilde{u}_i$, on $\partial\Omega_{u_i} \subseteq \partial\Omega$. Under the assumption of small displacement gradients, the infinitesimal strain tensor

$$\varepsilon_{ij} = u_{(i,j)} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{7.2}$$

is used as an appropriate measure for the deformation of the body. The Cauchy stress tensor, $\sigma(x) \in \mathbb{R}^{d_s \times d_s}$, is used as the corresponding stress measure. An external traction $\tilde{t}_i$ acts on the Neumann boundary $\partial\Omega_{t_i} \subseteq \partial\Omega$ and is equal to the

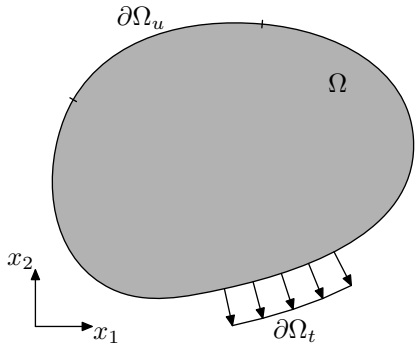


Figure 7.23: Solid domain Ω with boundary $\partial\Omega$.

projection of the stress tensor on the outward pointing normal vector $n(x) \in \mathbb{R}^N$, i.e. $t_i = \sigma_{ij} n_j$. The solid body is loaded by increasing the boundary tractions or boundary displacements. We refer to a stepwise increase of the boundary conditions as a load step.

7.3.1.1 Constitutive modeling

In isotropic continuum damage models, the Cauchy stress is related to the infinitesimal strain tensor by

$$\sigma_{ij} = (1 - \omega) C_{ijkl} \varepsilon_{kl}, \tag{7.3}$$

where $\omega \in [0, 1]$ is a scalar damage parameter and C is the Hookean elasticity tensor for undamage material (i.e. with $\omega = 0$). When damage has fully developed ($\omega = 1$) a material has lost all stiffness. Note that we adopt index notation with summation from 1 to d_s over repeated italic indices.

The damage parameter is related to a history parameter κ by a monotonically increasing function $\omega = \omega(\kappa)$, which is referred to as the damage law. The history parameter evolves according to the Kuhn-Tucker conditions

$$f \leq 0, \quad \dot{\kappa} \geq 0, \quad \dot{\kappa} f = 0 \tag{7.4}$$

for the loading function $f = \bar{\eta} - \kappa$, where $\bar{\eta}$ is a nonlocal strain measure, referred to as the nonlocal equivalent strain. The monotonicity of both κ and $\omega(\kappa)$ guarantees that the damage parameter is monotonically increasing at every material point, thereby introducing irreversibility in the constitutive model.

Nonlocality is introduced into the model by means of the nonlocal equivalent strain which ensures a well-posed formulation at the onset of damage evolution. If instead the damage parameter was related to a local strain measure, η, the resulting medium would suffer from a local loss of ellipticity in the case of material softening [110]. The model is then unable to smear out the damage zone over a finite volume. In other words, a local continuum damage formulation fails to introduce a length scale for the damage zone, resulting in spurious mesh dependencies in numerical solutions.

A straightforward way of introducing nonlocality in the formulation is by defining the nonlocal equivalent strain, $\bar{\eta}(x)$, as the volume average of a local equivalent strain, $\eta = \eta(\varepsilon)$,

$$\bar{\eta}(x) = \frac{\int\limits_{y \in \Omega} g(x,y)\eta(y)\,\mathrm{d}y}{\int\limits_{y \in \Omega} g(x,y)\,\mathrm{d}y}, \tag{7.5}$$

where $g(x,y)$ is the weighting function

$$g(x,y) = \exp\left(-\frac{\|x-y\|^2}{2l_c^2}\right). \tag{7.6}$$

We refer to this model as the nonlocal damage formulation [51]. The local equivalent strain maps the strain tensor onto a scalar. In the numerical simulations section we will employ various equivalent strain relations.

Although the nonlocal formulation is straightforward, it requires the computation of a volume integral for the evaluation of the constitutive behavior at every

material point. This makes the numerical implementation both cumbersome and inefficient. In particular, the stiffness matrix is full. Even when truncated, the nonlocal operator has a negative impact on the sparsity of the matrix. This results in computationally expensive assembly and solution routines. To circumvent these deficiencies, approximations of the integral equation are commonly used.

The nonlocal equivalent strain (7.5) can be approximated by an implicit gradient formulation (e.g. [86])

$$\bar{\eta}(x) - \frac{1}{2} l_{\mathrm{c}}^2 \frac{\partial^2 \bar{\eta}}{\partial x_i^2}(x) + \frac{1}{8} l_{\mathrm{c}}^4 \frac{\partial^4 \bar{\eta}}{\partial x_i^2 \partial x_j^2}(x) - \frac{1}{48} l_{\mathrm{c}}^6 \frac{\partial^6 \bar{\eta}}{\partial x_i^2 \partial x_j^2 \partial x_k^2}(x) + \ldots = \eta(x). \quad (7.7)$$

Because only C^0-continuity is required for the second-order approximation, the corresponding implicit gradient formulation has enjoyed widespread use.

We study the convergence of the implicit gradient formulation toward the nonlocal formulation upon increasing the number of gradient terms involved. If we truncate equation (7.7) after the d-th derivative, we can rewrite it using a linear operator $\mathcal{L}^d$ as

$$\mathcal{L}^d \bar{\eta}(x) = \eta(x). \quad (7.8)$$

We restrict ourselves to the second-order ($d = 2$), fourth-order ($d = 4$) and sixth-order ($d = 6$) implicit gradient damage formulations.

7.3.1.2 Implicit gradient damage formulation

In contrast to the nonlocal gradient damage formulation, the implicit formulation requires the solution of a boundary value problem for the nonlocal equivalent strain field, $\bar{\eta}(x)$, in addition to the usual problem for the displacement field, $u(x)$. In the absence of body forces, the resulting boundary value problem for the d-th

order formulation is given by

$$\begin{cases} \frac{\partial \sigma_{ij}}{\partial x_j} = 0 & \\ \mathcal{L}^d \bar{\eta} = \eta & \forall x \in \Omega \\ \sigma_{ij} n_j = \tilde{t}_i & \forall x \in \partial\Omega_{t_i} \\ \frac{\partial}{\partial x_n}\left(\frac{\partial^\alpha \bar{\eta}}{\partial x_{j\ldots}}\right) = 0 & \forall x \in \partial\Omega,\ \alpha \in \{0, \ldots, d-2\} \\ u_i = \tilde{u}_i & \forall x \in \partial\Omega_{u_i} \end{cases} \tag{7.9}$$

where $\tilde{t}$ and $\tilde{u}$ are the prescribed boundary traction and displacements, respectively. Notice that we assume all directional derivatives, $\frac{\partial}{\partial x_n} = n_i \frac{\partial}{\partial x_i}$, of the nonlocal equivalent strain field zero on the boundary. We verify this choice numerically by comparing the results with the nonlocal formulation based on the integral equation (7.5). The kinematic and constitutive relations (7.2) and (7.3) are used to express the Cauchy stress in terms of the displacement field. We solve the system (7.9) using the Galerkin method where the same solution spaces are used for the displacement field and nonlocal equivalent strain field, respectively.

7.3.2 Numerical results

We consider the L-shaped specimen shown in Figure 7.24. The problem set-up is inspired by [67], but has been modified to illustrate the capabilities of T-spline-based isogeometric analysis. The free rotation of the rigid end-plates is incorporated by means of linear constraints on the boundary control points, which is possible due to the fact that the basis functions on the corresponding boundaries can exactly represent all affine motions and in particular rigid rotations and translations. The diagonal failure zone resulting from the set-up requires local mesh refinements in that direction, which can be achieved using analysis-suitable T-splines.

In the undamaged state a linear isotropic material is considered with modulus of elasticity $E = 10\,\text{GPa}$ and Poisson's ratio $\nu = 0.2$. Plane stress conditions

are assumed. The local equivalent strain proposed in [41] is used

$$\eta(\varepsilon) = \frac{k-1}{2k(1-2\nu)} I_1(\varepsilon) + \frac{1}{2k}\sqrt{\left(\frac{k-1}{1-2\nu} I_1(\varepsilon)\right)^2 + \frac{12k}{(1+\nu)^2} J_2(\varepsilon)}, \qquad (7.10)$$

where $I_1(\varepsilon) = \varepsilon_{ii}$ and $J_2(\varepsilon) = \frac{1}{2}\varepsilon'_{ij}\varepsilon'_{ij}$ are the first invariant of the strain tensor and second invariant of the deviatoric strain tensor, $\varepsilon'_{ij} = \varepsilon_{ij} - \frac{1}{3}\varepsilon_{kk}\delta_{ij}$, respectively. Note that in the plane stress case, the summations in the expressions for the strain invariants are taken from 1 to 3. The parameter $k = 10$ distinguishes the cases of tension and compression. The following damage law as proposed in [52] is used

$$\omega(\kappa) = \begin{cases} 0 & \kappa \leq \kappa_0 \\ 1 - \frac{\kappa_0}{\kappa}\left\{(1-\alpha) + \alpha \exp\left[\beta(\kappa_0 - \kappa)\right]\right\} & \kappa > \kappa_0 \end{cases} \qquad (7.11)$$

with parameters $\kappa_0 = 4 \cdot 10^{-4}$, $\alpha = 0.98$ and $\beta = 80$. The nonlocal length scale is taken as $l_c = 5\sqrt{2} \approx 7.07$ mm.

Force-displacement curves are obtained using the cubic Bézier meshes shown in Figure 7.25. A summary of the mesh parameters is given in Table 7.2. Note that for degree three T-splines the number of basis functions is similar to the number of elements, this in contrast to traditional cubic finite elements. A C^2-continuous base mesh is created using a non-tensor product T-spline. The C^2 basis function centered around the reentrant corner is shown in Figure 7.26. Meshes 1, 2 and 3 are obtained by subsequent analysis-suitable local refinements of a band along the symmetry plane. Mesh 4 is obtained as a global refinement of Mesh 3. The control point weights are all taken equal to 1. Displacement control is used to trace the equilibrium path.

The force-displacement curves obtained using the various Bézier meshes are shown in Figure 7.27. For the gradient formulations, the results obtained on Mesh 3 cannot be visually distinguished from those obtained using Mesh 4. Better convergence behavior is obtained by increasing the order of the formulation. Because

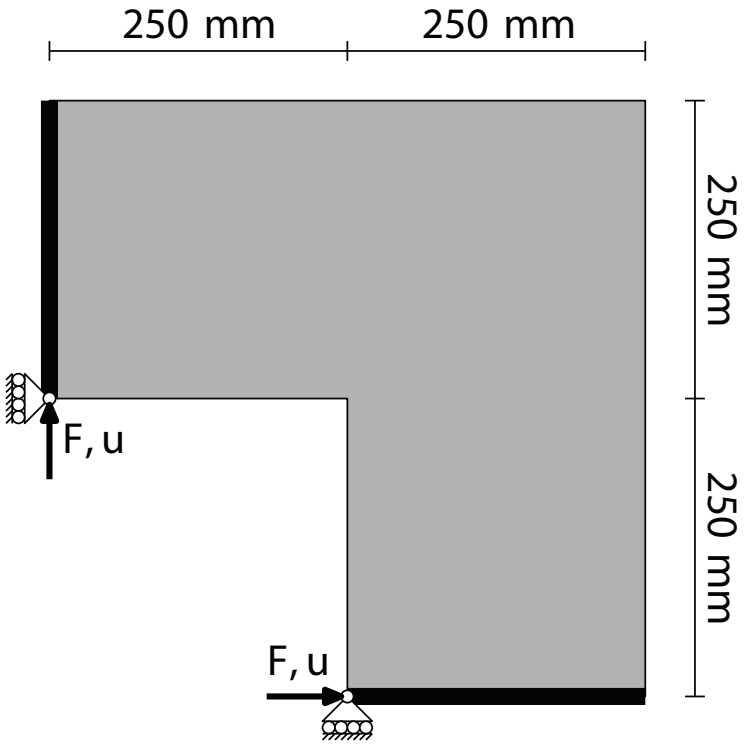
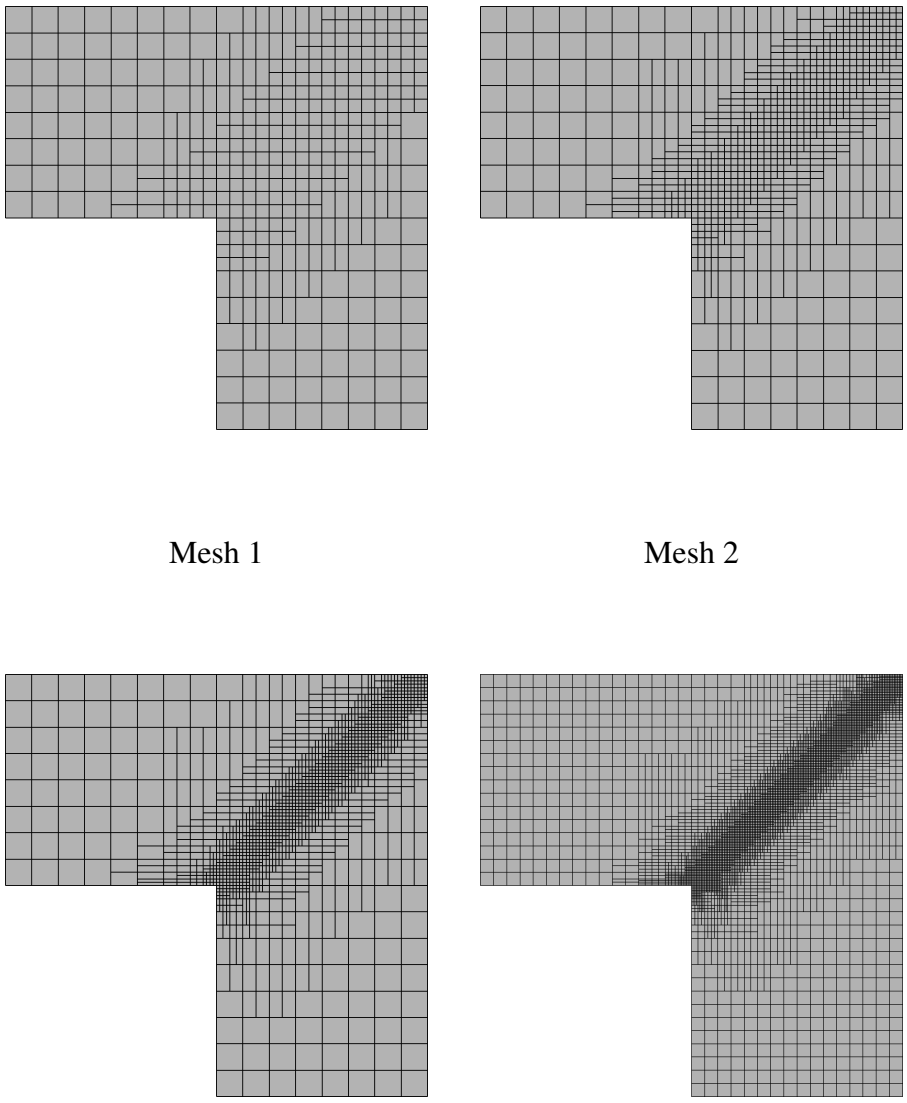


Figure 7.24: L-shaped specimen. The thickness of the specimen is 200 mm.

	Mesh 1	Mesh 2	Mesh 3	Mesh 4
Degree, p	3	3	3	3
Number of elements, n_e	391	816	1686	6032
Number of basis functions, n	473	832	1543	5714

Table 7.2: Bézier meshes used for the L-shaped specimen.



Mesh 1 Mesh 2

Mesh 3 Mesh 4

Figure 7.25: Bézier meshes for the L-shaped specimen.

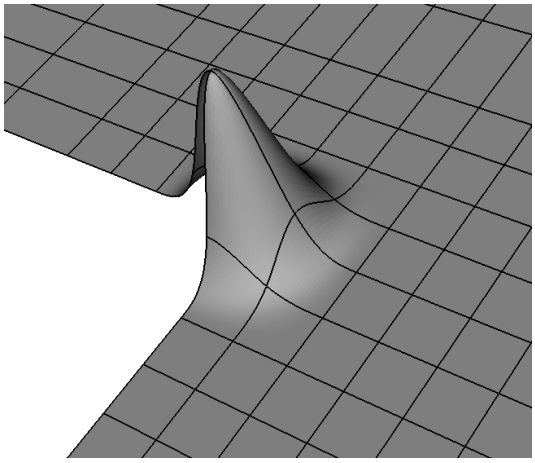


Figure 7.26: Smooth (C^2) T-spline basis function centered around the reentrant corner of the L-shaped domain.

of the involved computational effort, the force-displacement curves for the nonlocal formulation are obtained only on Meshes 1, 2 and 3. The force-displacement curve obtained on Mesh 3 coincides with that found on Mesh 2. For all formulations, the accuracy of the result obtained on Mesh 3 is sufficient to allow for comparison of the various formulations.

In Figure 7.28 the results of the various formulations are compared. Upon increasing the order of the formulation the approximation of the nonlocal result is improved. Increasing the order of the formulation increases the total amount of dissipated energy. This is caused by the additional smoothing effect of the higher-order derivatives, which can also be seen from the damage isolines in Figure 7.29. For the considered problem, the sixth-order formulation is observed to be very efficient, since it accurately approximates the nonlocal result, whereas the involved computational effort is negligible compared to the nonlocal formulation. A contour plot of the sixth-order damage corresponding to Mesh 3 is shown in Figure 7.30. It is observed that setting all the Neumann boundary conditions (7.9) for the equivalent

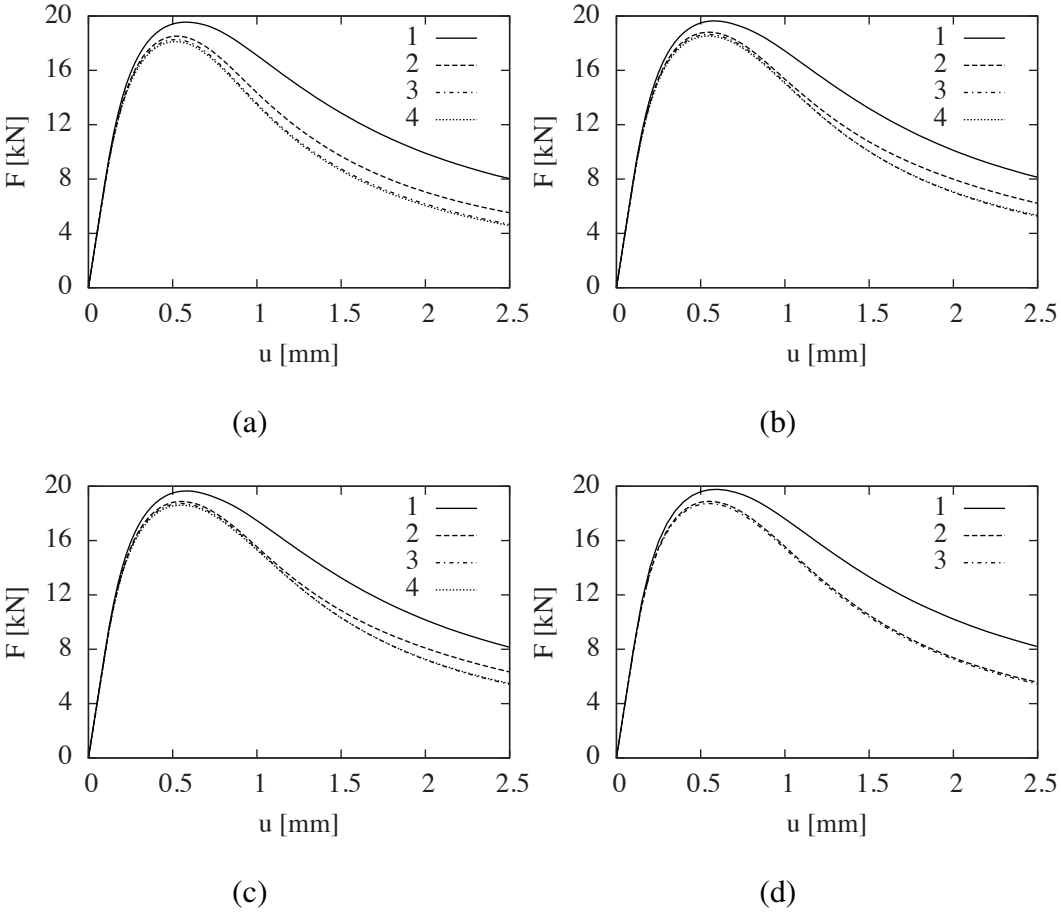


Figure 7.27: Mesh convergence studies using the cubic T-spline meshes in Figure 7.25 for the second-order (a), fourth-order (b) and sixth-order (c) damage formulations, and for the nonlocal formulation (d).

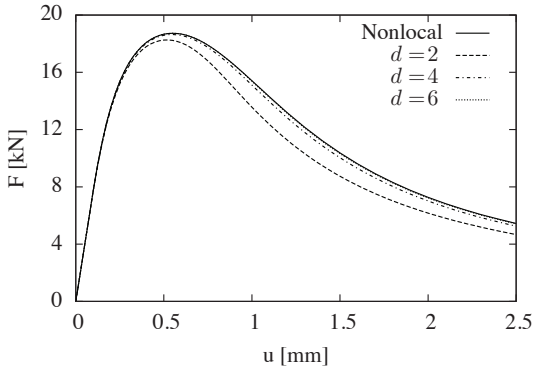


Figure 7.28: Force-displacement results for the L-shaped specimen using the nonlocal formulation and d-th order gradient formulations. All results are obtained using Mesh 3.

strain field to zero does not have a significant effect on the results.

7.4 Isogeometric phase-field fracture

The prevention of fracture-induced failure is a major constraint in engineering designs, and the numerical simulation of fracture processes often plays a key role in design decisions. Most methods represent cracks as discrete discontinuities which must be tracked numerically [6, 83]. Tracing the evolution of complex crack patterns has, however, proven to be a tedious task, particularly in three dimensions. As a consequence, a generally accepted numerical strategy for three-dimensional structures with complex crack patterns is still unavailable.

In this example, we simulate brittle fracture using a phase-field [25, 81, 82], analysis-suitable T-splines, Bézier extraction, and local refinement. Using this approach, discontinuities are modeled by a phase-field and the fracture surfaces do not need to be tracked numerically. To efficiently resolve the small crack length scales requires robust local and adaptive refinement strategies and element technol-

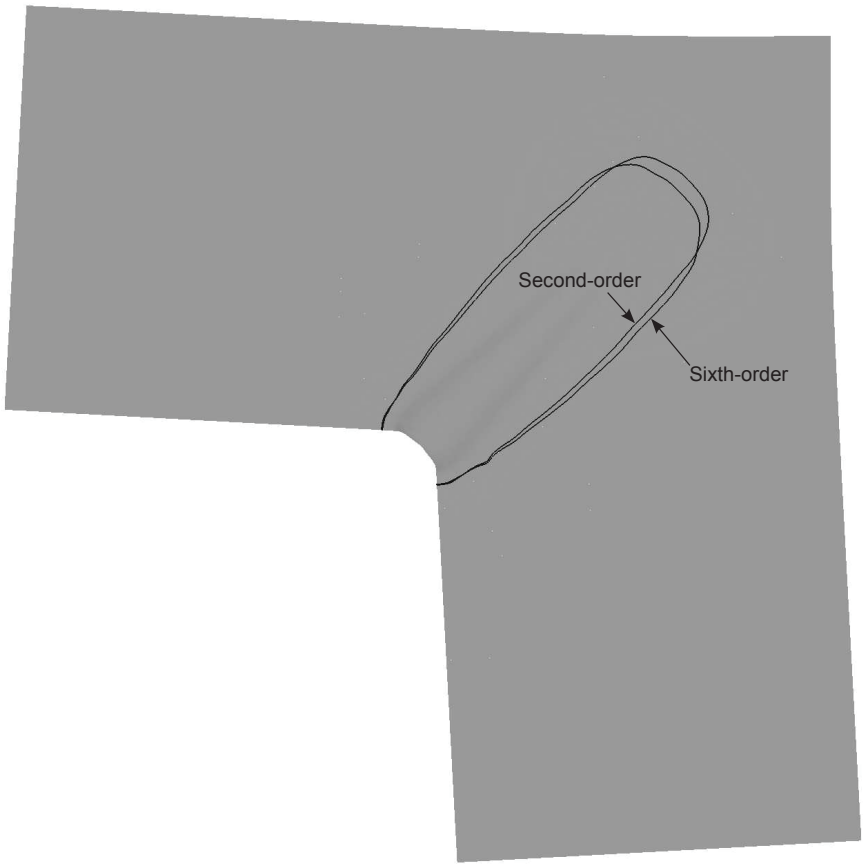


Figure 7.29: Isolines for the damage parameter $\omega = 0.8$ in the L-shaped specimen at $u = 2\,\text{mm}$ as computed on Mesh 3 by the second-order formulation and the sixth-order formulation. Displacements are amplified by a factor of 15.

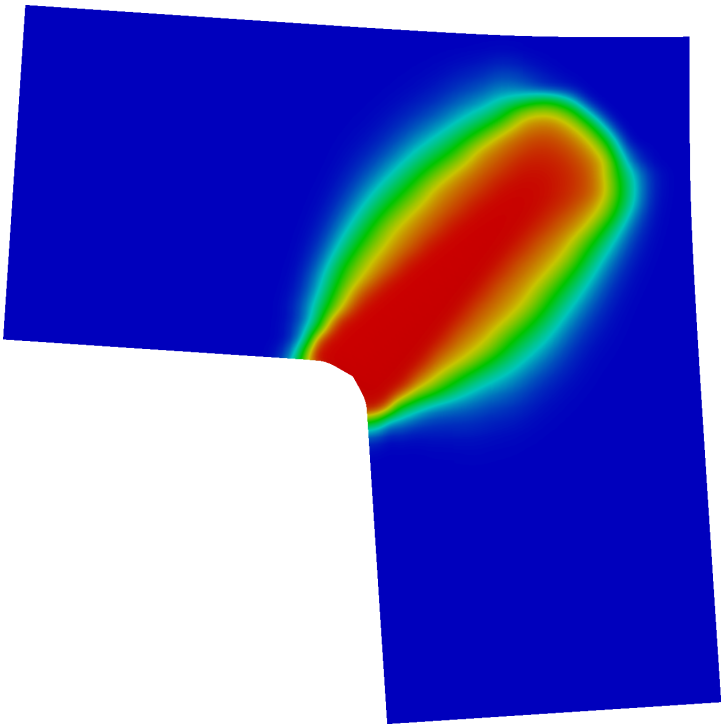


Figure 7.30: A contour plot of the sixth-order damage field corresponding to Mesh 3.

ogy which can be employed in a parallel computing environment. We demonstrate the potential of T-splines to handle these requirements.

Since our intent is to demonstrate the key features of T-splines, we only briefly describe the main ideas of the phase-field fracture formulation. For a thorough treatment we refer the interested reader to [24].

7.4.1 Isotropic fracture formulation

We consider an arbitrary body $\Omega \subset \mathbb{R}^{d_s}$ (with $d_s \in \{1, 2, 3\}$) with external boundary $\partial\Omega$ and internal discontinuity boundary Γ (see Figure 7.31a), which rep-

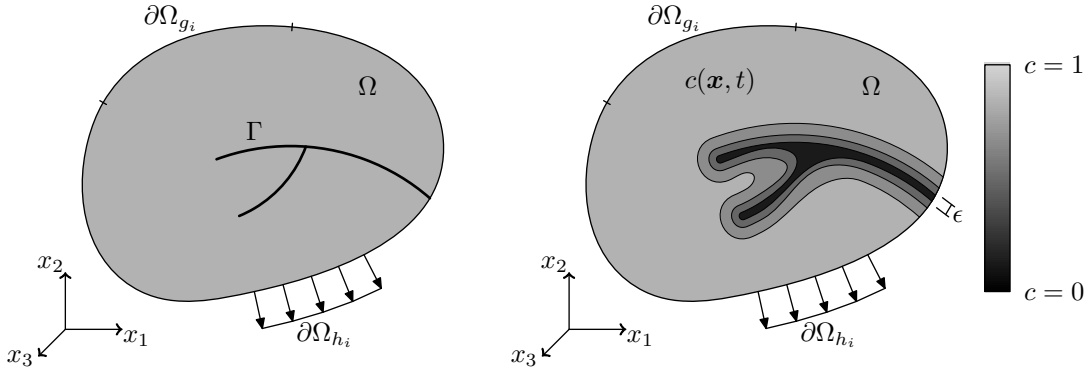
resents a set of discrete cracks. The displacement of a point $\boldsymbol{x} \in \Omega$ at time $t \in [0, T]$ is denoted by $\boldsymbol{u}(\boldsymbol{x}, t) \in \mathbb{R}^{d_s}$. Spatial components of vectors and tensors are indexed by $i, j = 1, \ldots, d$. The displacement field satisfies time-dependent Dirichlet boundary conditions, $u_i(\boldsymbol{x}, t) = g_i(\boldsymbol{x}, t)$, on $\partial\Omega_{g_i} \subseteq \partial\Omega$, and time-dependent Neumann boundary conditions on $\partial\Omega_{h_i} \subseteq \partial\Omega$. We assume small deformations and deformation gradients, and define the infinitesimal strain tensor, $\boldsymbol{\varepsilon}(\boldsymbol{x}, t) \in \mathbb{R}^{d_s \times d_s}$, with components

$$\varepsilon_{ij} = u_{(i,j)} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{7.12}$$

as an appropriate deformation measure. We assume isotropic linear elasticity, such that the elastic energy density is given by

$$\psi_e(\boldsymbol{\varepsilon}) = \frac{1}{2}\lambda\varepsilon_{ii}\varepsilon_{jj} + \mu\varepsilon_{ij}\varepsilon_{ij} \tag{7.13}$$

with λ and μ the Lamé constants. Note that we use the Einstein summation convention on repeated indices.



(a) (b)

Figure 7.31: (a) Schematic representation of a solid body Ω with internal discontinuity boundaries Γ. (b) Approximation of the internal discontinuity boundaries by the phase-field $c(x, t)$. The model parameter ϵ controls the width of the failure zone.

In order to circumvent the problems associated with numerically tracking the propagating discontinuity representing a crack, we approximate the fracture surface, Γ, by a phase-field, $c(\boldsymbol{x}, t) \in [0, 1]$. The value of this phase-field is equal to 1 away from the crack and is equal to 0 inside the crack (see Figure 7.31b).

7.4.2 Adaptive refinement scheme

As shown in Figure 7.31b, the length scale parameter, ϵ, plays two roles in the phase-field model: first, it determines the width of the approximation to the crack, and second, as shown in [24], it influences the magnitude of the tensile stress required for crack nucleation. Thus, in order to capture fine scale details of a crack, or model materials with high nucleation stresses, a small value for ϵ is needed. This in turn requires a fine mesh in areas where the crack is located.

To efficiently compute with the fine mesh as needed to accurately resolve a crack for small values of ϵ, we introduce an adaptive refinement scheme. For this scheme we choose the phase-field parameter as a convenient measure for determining the need for refinement. The gradients of the phase-field are high in an area near the crack. Away from the crack the value of the phase-field stays close to one. By choosing a critical threshold of the phase-field that is higher than the value at which crack nucleation occurs ($c = 0.75$) the area near the crack is easily identified. Using a larger value for the critical threshold results in a greater area of refinement (we have found $c = 0.8$ to be a good choice). The adaptive refinement scheme we have developed proceeds as follows:

1. Run the dynamic simulation to some termination point
2. Flag elements where the phase-field is below the critical threshold
3. Refine the flagged elements

4. Rerun the simulation with the locally refined mesh

5. Repeat steps 2—4 until convergence

7.4.3 Numerical results

We investigate the numerical performance of the phase-field fracture model using T-spline spatial discretizations and adaptive local refinement as described in Chapter 5.6. To integrate over the elements we use Gaussian quadrature with a $p + 1$ rule where p is the polynomial degree of the basis functions. Thus, in two-dimensions we use a 3-by-3 quadrature rule for quadratic basis functions and a 4-by-4 quadrature rule for cubic basis functions.

For the examples below, the reported mesh sizes, h, are computed on the Bézier elements as $h = \sqrt[d_s]{a}$ where a is the area of an element in two dimensions and the volume of an element in three dimensions and d_s is the number of spatial dimensions. In most cases, the mesh is such that $h = \epsilon/2$ in the area where a crack has formed. Experience has shown that this relationship between h and ϵ provides sufficient accuracy without over resolving the crack.

7.4.3.1 Dynamic shear loading

In this example we model crack initiation and propagation under a dynamic shear load. The model is based on experimental results reported by [63] and [62]. Previous numerical results of this problem based on XFEM have been reported by [16], among others, and a comparison of results between XFEM, the element deletion method, and the interelement crack method have been reported by [111]. Numerical results of a similar experiment reported by [92] have been reported by [95] where cohesive segments have been used to model the crack.

The input geometry and loading conditions for the simulation are shown in Figure 7.32, where symmetry is employed to reduce the computational cost. In the experiment, the load was applied by firing a projectile at a prenotched specimen. In our simulation we model the case where the projectile was fired with a velocity of $33\,\mathrm{m/s}$ by applying the kinematic velocity

$$v = \begin{cases} \frac{t}{t_0} v_0 & t \le t_0 \\ v_0 & t > t_0 \end{cases} \tag{7.14}$$

where $v_0 = 16.5\,\mathrm{m/s}$ and $t_0 = 1\mu\mathrm{s}$. A no traction boundary condition is applied to all unspecified surfaces. The initial crack is modeled by a discontinuity in the geometry in order to introduce a sharp crack tip.

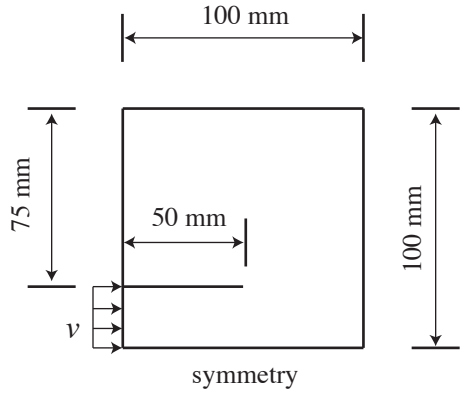


Figure 7.32: The geometry and boundary conditions for the dynamic shear loading example. The crack is modeled by an actual discontinuity in the mesh with a zero radius crack tip. The load is applied as a velocity condition that is ramped up from 0 to 16.5 m/s in one microsecond and then held constant for the duration of the simulation.

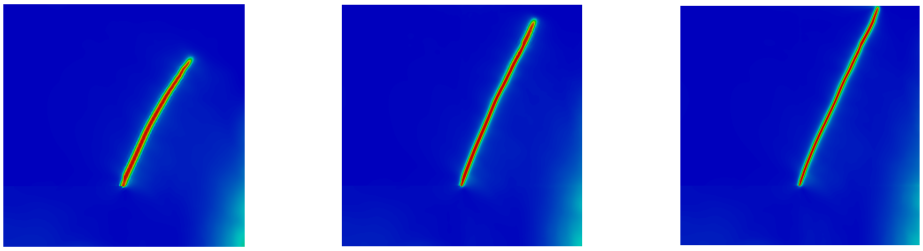
The model parameters are $\rho = 8000\,\mathrm{kg/m^3}$, $E = 190\,\mathrm{GPa}$, $\nu = 0.3$, $\mathcal{G}_c = 2.213 \times 10^4\,\mathrm{J/m^2}$, $k = 0$, and plane strain is assumed. The corresponding dilatational, shear, and Rayleigh wave speeds are $v_d = 5654\,\mathrm{m/s}$, $v_s = 3022\,\mathrm{m/s}$,

$v_R = 2803$ m/s. The length scale was chosen to be $\epsilon = 1.95 \times 10^{-4}\,\mathrm{m}$ leading to a maximum uniaxial stress of $1.07\ \mathrm{GPa}$.

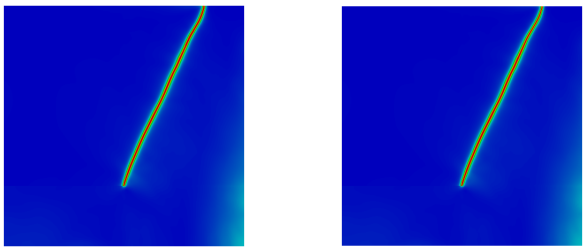
We start with a coarse initial C^2-continuous cubic T-spline that has 128×128 Bézier elements. For all meshes, ϵ is set to $1.95 \times 10^{-4}\,\mathrm{m}$. Elements are flagged for refinement if the phase-field parameter is less than 0.8 at any quadrature point within the element. The sequence of results shown in Figure 7.33 where each simulation was terminated at $t = 100\,\mu s$. Figure 7.34 shows the sequence of meshes with the elements that have been flagged for refinement at the end of each iteration. Note that when the mesh is too coarse, the crack propagation is restricted and the direction is incorrect. It is not until mesh 3, when $h = \epsilon$, that the mesh is fine enough to capture the correct crack path.

Figure 7.35 compares the elastic strain energy and dissipated energy at each refinement iteration to a reference solution computed with a uniform quadratic NURBS mesh with 1024×1024 Bézier elements. The elastic strain energy, shown in Figure 7.35(a), is over predicted for the coarse meshes as a result of restricted crack propagation. This plot shows that it is not until mesh 4, when most of the element along the propagation path are such that $h = \epsilon/2$, that the elastic strain energy agrees well with the reference solution. This is also true for the dissipated energy shown in Figure 7.35(b).

Table 7.3 lists the total number of functions for each mesh in the refinement sequence and the number of elements that were flagged at the end of each simulation. Note that the final mesh has 53,032 cubic basis functions. This is compared to 1,055,242 quadratic basis functions in the uniformly refined reference solution.



Mesh 1 Mesh 2 Mesh 3



Mesh 4 Mesh 5

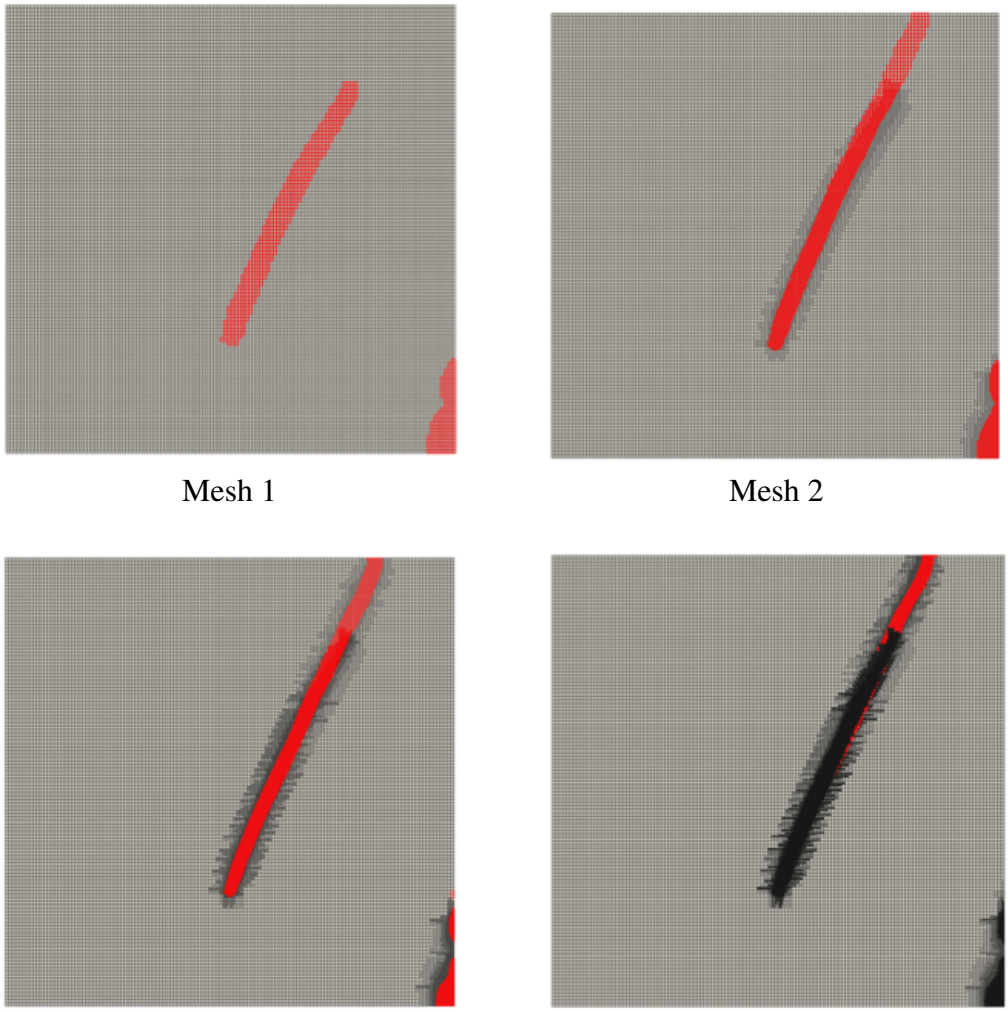
Figure 7.33: Kalthoff mesh refinement results. Mesh 1 is a 128 x 128 cubic T-spline mesh. Bézier elements were flagged for refinement if $c < 0.8$ at any quadrature point inside the element and $h = \sqrt{a} > 1.94 \times 10^{-4}$ m where a is the element area.

7.4.3.2 Pressurized cylinder with solid elements

A major benefit of the phase-field formulation presented here is that it extends easily to three dimensions. As a final example, we show a three-dimensional computation of a pressurized cylinder with a spherical end cap. The input geometry for the simulation is shown in Figure 7.36 where symmetry is used to reduce the computational cost. The initial crack is modeled by an induced phase-field

	Mesh 1	Mesh 2	Mesh 3	Mesh 4	Mesh 5
Number of functions	17,755	19,992	27,032	47,824	53,032
Flagged elements	589	2,001	6,257	1,446	8

Table 7.3: The number basis functions before refinement and the number of elements that were flagged for refinement for each mesh.



Mesh 1 Mesh 2

Mesh 3 Mesh 4

Figure 7.34: The first four meshes in the local refinement sequence. The elements in red are those that were selected to be refined at each step.

(see [24]). A linearly increasing hydrostatic pressure load, p, is applied to the inner surface as $p = 50t\,\mathrm{MPa}$ where t is the current time.

The model parameters are $\rho = 8000$ kg/m^3, $E = 190$ GPa, $\nu = 0.3$, $\mathcal{G}_c =$

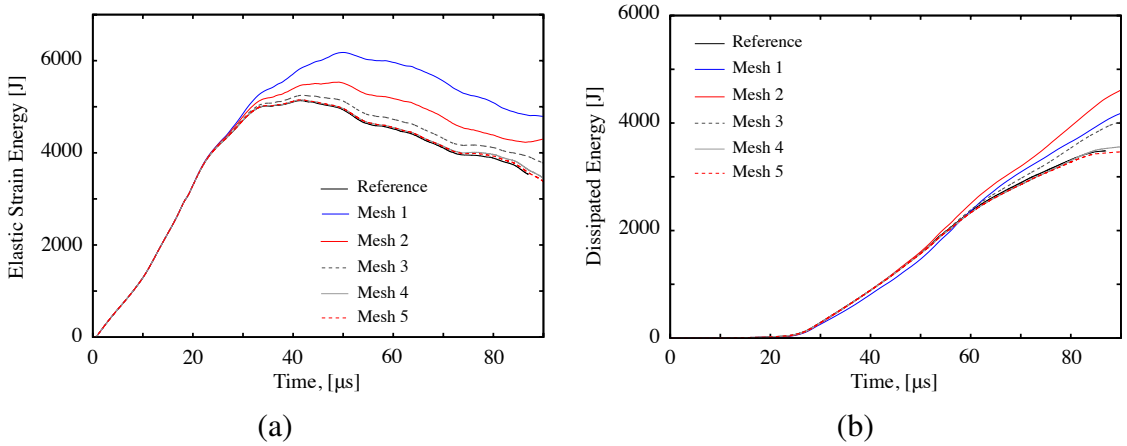


Figure 7.35: The (a) elastic strain energy and (b) dissipated energy for the sequence of refined meshes shown in Figure 7.33. The reference mesh is a uniform quadratic NURBS mesh with 1024×1024 Bézier elements.

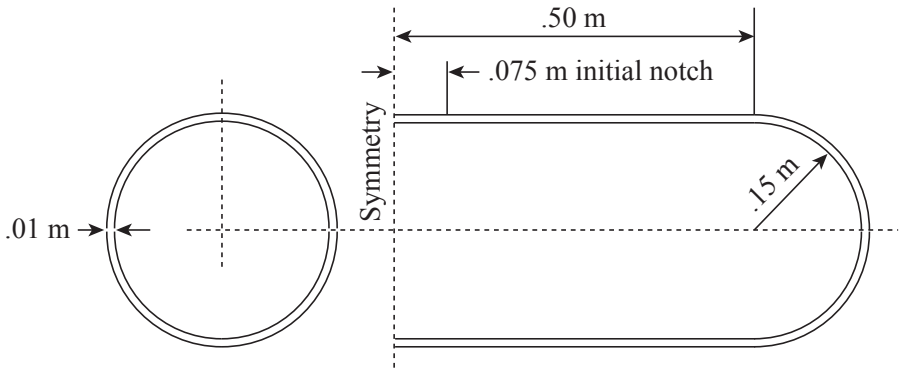


Figure 7.36: Geometry and symmetry conditions for the pressure vessel simulation. The mesh is a three-dimensional thickened T-spline.

2.213×10^4 J/m^2, and $k = 0$. The corresponding dilatational, shear, and Rayleigh wave speeds are $v_d = 5654$ m/s, $v_s = 3022$ m/s, $v_R = 2803$ m/s. The length scale was chosen to be $\epsilon = 2.5 \times 10^{-3}$ m leading to a maximum uniaxial stress of 298 MPa (see [24]).

To construct the mesh, a cubic T-spline mid-surface was first modeled in

Rhino, a commercial CAD software package, using the T-Splines plugin. Note that there are four valence three extraordinary points present in the end cap. The initial mid-surface mesh had a mesh size of $h \approx 0.01\,\mathrm{m}$. After export, the surface was thickened with C^1-continuous quadratic functions such that there were eight Bézier elements (eleven functions) through the thickness. To get the final mesh, we used the adaptive refinement scheme describe in Section 7.4.2 . The refinement was applied to the mid-surface mesh at each iteration until $h \approx \epsilon/2$ in the area of the crack. A new volume mesh was created from the updated mid-surface mesh at each iteration. The final mesh is shown in Figure 7.37. This mesh contains 862,100 basis functions.

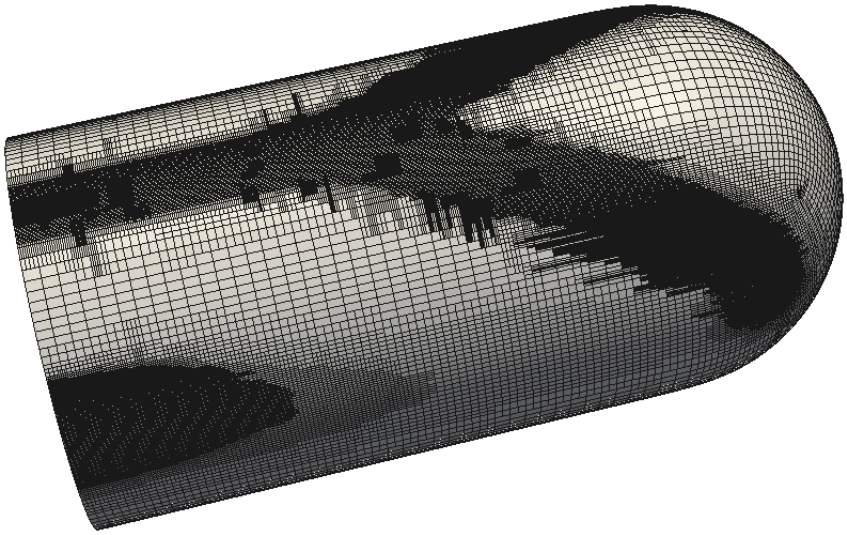
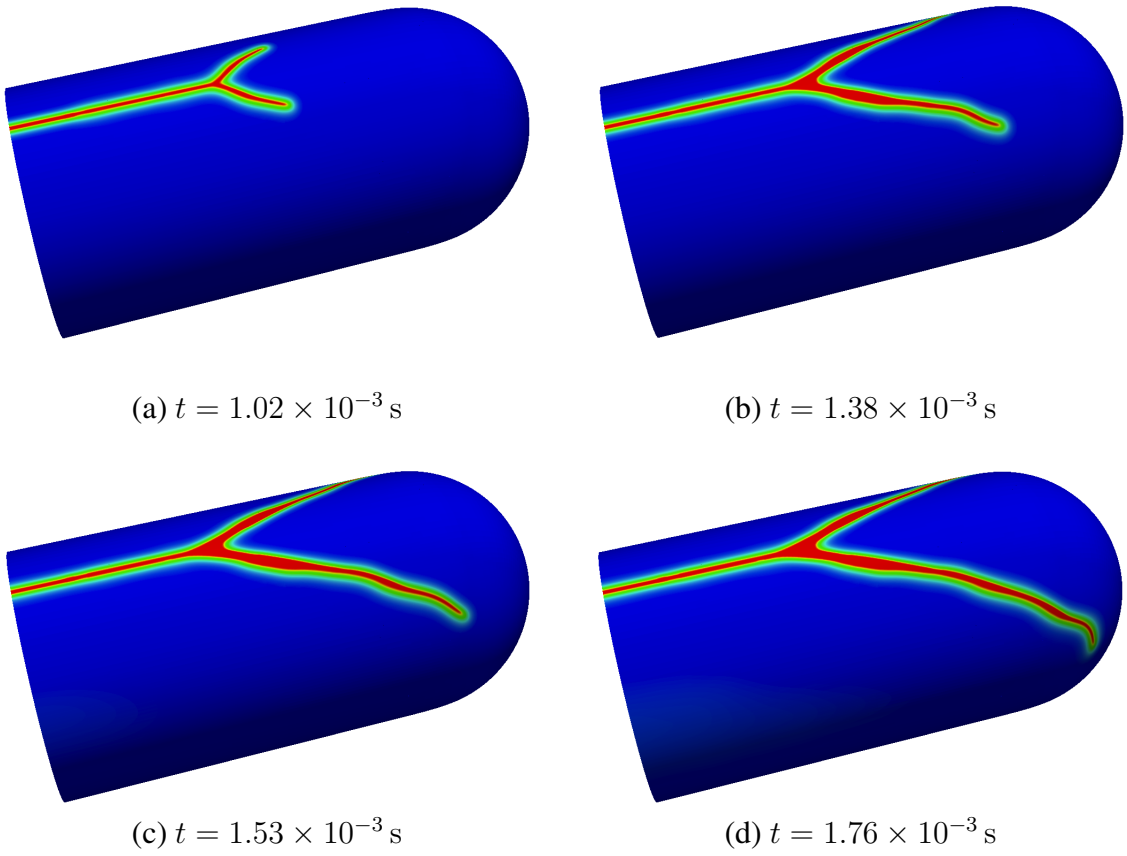


Figure 7.37: The final mesh for the pressurized cylinder example problem. The volumetric mesh was constructed by thickening a mid-surface mesh. The refinement was performed using the adaptive refinement scheme describe in Section 7.4.2 which resulted in a final mesh containing 862,100 basis functions.

The resulting phase-field is shown at several time intervals in Figure 7.38. A post-processed plot of the model at $t = 1.76 \times 10^{-3}\,\mathrm{s}$ is shown in Figure 7.39 with

the displacements scaled by a factor of 5 and the area of the model where $c < 0.05$ removed from the visualization.



(a) $t = 1.02 \times 10^{-3}\,\mathrm{s}$ (b) $t = 1.38 \times 10^{-3}\,\mathrm{s}$

(c) $t = 1.53 \times 10^{-3}\,\mathrm{s}$ (d) $t = 1.76 \times 10^{-3}\,\mathrm{s}$

Figure 7.38: The results of the pressurized cylinder example. The phase-field is shown.

Figure 7.40 shows two cross-section views of the crack at different times in the simulation. These cross-sections show the ability of the phase-field model to capture three-dimensional characteristics of a crack. We emphasize that the computation for this three-dimensional model did not require any additional algorithmic complexity compared to the two-dimensional models shown previously.

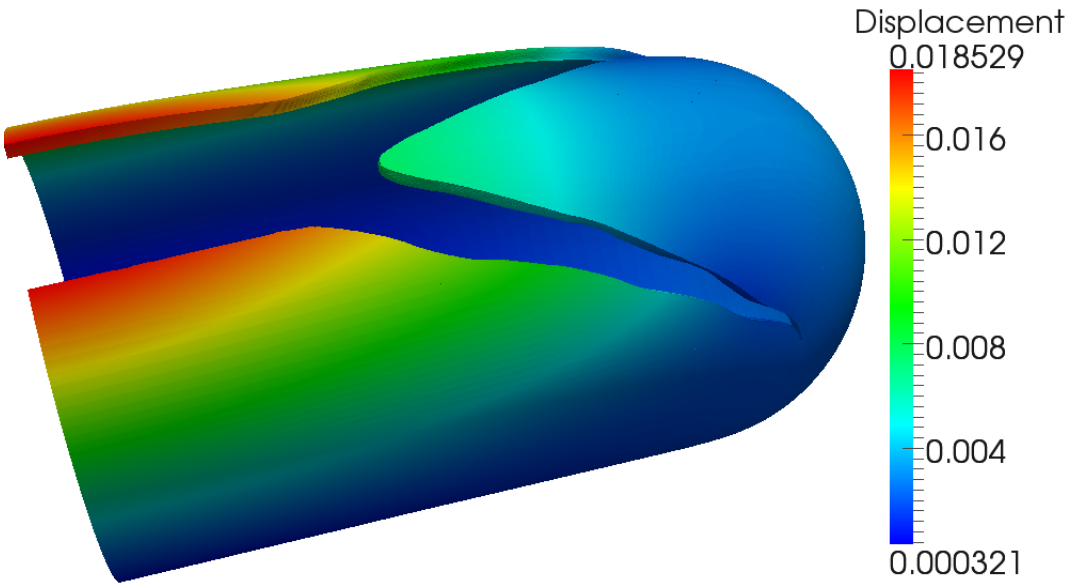


Figure 7.39: A post-processed plot of the pressure vessel example at $t = 1.76 \times 10^{-3}\,\mathrm{s}$. The displacements have been scaled by a factor of 5 and areas of model where $c < 0.05$ have been removed from the plot. Displacement is measured in meters.

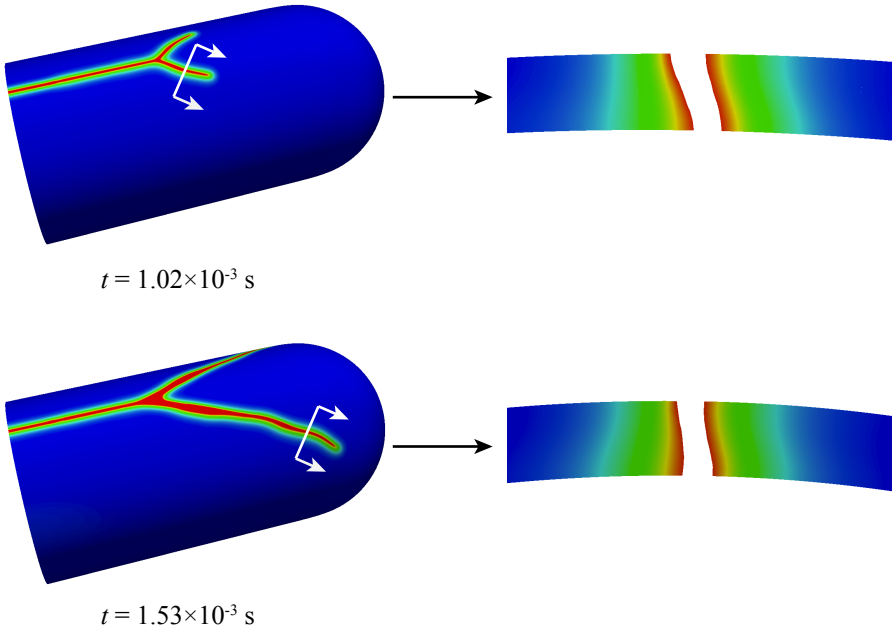


Figure 7.40: Cross section views showing the three-dimension profile of the crack surfaces.

Chapter 8

Conclusions and future work

In this dissertation, a design-through-analysis framework based on T-splines and isogeometric analysis is developed. Specifically, we focus on bicubic T-spline surfaces. In all cases, the technology meets the demands of both design *and* analysis. We have demonstrated that T-splines are an ideal discretization technology for isogeometric analysis and, on a higher level, a foundation upon which unified DTA technologies can be built. The main contributions can be summarized as follows:

- Development of Bézier extraction as a unifying paradigm underlying isogeometric finite element technology.
- Characterization of analysis-suitable T-splines in analysis.
- Formulation and implementation of a localized analysis-suitable refinement algorithm.
- Formulation and implementation of analysis-suitable T-splines which accommodate extraordinary points (i.e., unstructured meshes).
- Development and implementation of an efficient adaptive isogeometric framework which couples analysis-suitable T-splines, Bézier extraction, and local refinement, and application of this strategy to problems of implicit gradient damage and phase-field modeling of brittle fracture.

There exist many promising future research directions on the geometric and analysis aspects of T-spline-based isogeometric analysis. On the geometric side, T-splines of arbitrary degree need to be fully developed with an analogue of analysis-suitability identified for even polynomial degrees. Additionally, degree elevation techniques must be established and coupled with existing local h-refinement schemes. Better techniques for handling non-uniform knot intervals around extraordinary points must be identified. These techniques should not produce excessive oscillations when widely varying knot intervals are present in the T-mesh. Perhaps the most challenging problem, at least from an implementational point-of-view, is the development of a fully trivariate version of T-splines which includes extraordinary points. Fortunately, it appears that most of the theory and practice developed in this dissertation can be applied in the trivariate setting with little modification. Parallelizing both storage and representation schemes as well as fundamental T-spline algorithms such as local refinement will be essential to accomodate multi-million control point T-meshes in design and analysis.

On the analysis side, there are many areas of application where T-spline discretizations would advance the state of the art. A sampling of these applications could include: plates and shells, fluids and fluid-structure interaction, contact, shape optimization, and electromagnetics. There are many others which could be named. In each case, T-splines could provide unprecendented accuracy and robustness when compared to standard finite elements. Additionally, the superior modeling capabilities of T-splines would allow these isogeometric application domains to be extended to more complex, real-world scenarios in a fully integrated DTA environment. On a more fundamental note, efficient quadrature schemes which account for T-junctions and extraordinary points must be developed. Also, basic approximation, stability, and error estimates must be established for analysis-suitable T-spline

spaces. Additionally, extending the refinement capabilities of T-splines to accomodate analysis-suitable hierarchical approaches on unstructured meshes may be advantageous to analysis [50].

Appendices

Appendix A

T-spline extraction example data

A.1 T-spline control points

Table A.1 lists the global T-spline control point coordinates for the geometry in Figure 3.1.

A.2 Bézier element control points

Table A.2 lists the Bézier element control points for elements 1, 10, 11, and 17 from the geometry in Figure 3.1.

Control point	x	y	w
1	0.0000	1.5000	1.0000
2	0.1858	1.5000	0.9512
3	0.5746	1.4288	0.8780
4	1.0022	1.1497	0.8475
5	1.2637	0.8275	0.8597
6	1.4477	0.4714	0.8963
7	1.5000	0.1858	0.9512
8	1.5000	0.0000	1.0000
9	0.0000	1.6250	1.0000
10	0.2013	1.6250	0.9512
11	0.6224	1.5479	0.8780
12	1.0857	1.2455	0.8475
13	1.3690	0.8964	0.8597
14	1.5683	0.5107	0.8963
15	1.6250	0.2013	0.9512
16	1.6250	0.0000	1.0000
17	0.0000	1.8750	1.0000
18	0.2323	1.8750	0.9512
19	0.7182	1.7860	0.8780
20	1.2527	1.4371	0.8475
21	1.5270	0.9999	0.8597
22	1.7493	0.5696	0.8963
23	1.8425	0.2246	0.9512
24	1.8425	0.0000	1.0000
25	1.7376	1.1378	0.8597
26	1.9906	0.6482	0.8963
27	2.0625	0.2555	0.9512
28	2.0625	0.0000	1.0000
29	0.0000	2.2500	1.0000
30	0.2788	2.2500	0.9512
31	0.8618	2.1432	0.8780
32	1.5033	1.7245	0.8475
33	1.9516	1.2194	0.7895
34	2.2319	0.7268	0.8963
35	2.3125	0.2865	0.9512
36	2.3125	0.0000	1.0000
37	0.0000	2.6250	1.0000
38	0.3252	2.6250	0.9512
39	1.0055	2.5004	0.8780
40	1.9100	1.9100	0.8413
41	2.5004	1.0055	0.8780
42	2.6250	0.3252	0.9512
43	2.6250	0.0000	1.0000
44	0.0000	2.8750	1.0000
45	0.3562	2.8750	0.9512
46	1.1012	2.7386	0.8780
47	2.0919	2.0919	0.8413
48	2.7386	1.1012	0.8780
49	2.8750	0.3562	0.9512
50	2.8750	0.0000	1.0000
51	0.0000	3.0000	1.0000
52	0.3717	3.0000	0.9512
53	1.1491	2.8576	0.8780
54	2.1829	2.1829	0.8413
55	2.8576	1.1491	0.8780
56	3.0000	0.3717	0.9512
57	3.0000	0.0000	1.0000

Table A.1: The control point ($\mathbf{P}$) coordinates (x, y) and weights (w).

$a(i,j)$	x	y	w
1	0.5521	1.3947	0.8902
2	0.5982	1.5109	0.8902
3	0.6442	1.6271	0.8902
4	0.6902	1.7433	0.8902
5	0.3724	1.4658	0.9146
6	0.4035	1.5880	0.9146
7	0.4345	1.7101	0.9146
8	0.4655	1.8323	0.9146
9	0.1858	1.5000	0.9512
10	0.2013	1.6250	0.9512
11	0.2168	1.7500	0.9512
12	0.2323	1.8750	0.9512
13	0.0000	1.5000	1.0000
14	0.0000	1.6250	1.0000
15	0.0000	1.7500	1.0000
16	0.0000	1.8750	1.0000

$e = 1$

$a(i,j)$	x	y	w
1	1.8750	0.0000	1.0000
2	1.9375	0.0000	1.0000
3	2.0000	0.0000	1.0000
4	2.0625	0.0000	1.0000
5	1.8750	0.2323	0.9512
6	1.9375	0.2401	0.9512
7	2.0000	0.2478	0.9512
8	2.0625	0.2555	0.9512
9	1.8323	0.4655	0.9146
10	1.8934	0.4810	0.9146
11	1.9544	0.4966	0.9146
12	2.0155	0.5121	0.9146
13	1.7433	0.6902	0.8902
14	1.8015	0.7132	0.8902
15	1.8596	0.7362	0.8902
16	1.9177	0.7592	0.8902

$e = 10$

$a(i,j)$	x	y	w
1	1.4584	1.4584	0.8536
2	1.5026	1.5026	0.8536
3	1.5468	1.5468	0.8536
4	1.5910	1.5910	0.8536
5	1.2570	1.6598	0.8536
6	1.2951	1.7101	0.8536
7	1.3332	1.7604	0.8536
8	1.3713	1.8107	0.8536
9	1.0202	1.8143	0.8658
10	1.0512	1.8693	0.8658
11	1.0821	1.9243	0.8658
12	1.1130	1.9793	0.8658
13	0.7592	1.9177	0.8902
14	0.7822	1.9758	0.8902
15	0.8052	2.0339	0.8902
16	0.8282	2.0920	0.8902

$e = 11$

$a(i,j)$	x	y	w
1	1.8832	1.2312	0.8627
2	1.9879	1.2996	0.8627
3	2.0925	1.3680	0.8627
4	2.1971	1.4364	0.8627
5	1.7985	1.3608	0.8566
6	1.8985	1.4364	0.8566
7	1.9984	1.5120	0.8566
8	2.0983	1.5876	0.8566
9	1.7008	1.4811	0.8536
10	1.7953	1.5634	0.8536
11	1.8898	1.6457	0.8536
12	1.9843	1.7280	0.8536
13	1.5910	1.5910	0.8536
14	1.6794	1.6794	0.8536
15	1.7678	1.7678	0.8536
16	1.8561	1.8561	0.8536

$e = 17$

Table A.2: Bézier element control points for elements 1, 10, 11 and 17, as shown in Figure 3.10. The array $a(i,j)$ is defined in (4.9).

A.3 Bézier element extraction operators

The extraction operators for elements 1, 10, 11 and 17 are listed below. These operators can be used to retrieve the Bézier control points listed in Appendix A.2 from the global T-spline control points in Appendix A.1 using (4.31). We note that because most extraction operators have many zero entries it is important to employ a sparse matrix storage format (see [97] and references therein.)

$$\mathbf{C}^1 = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.0000 & 0.0000 & 0.0000 \\
0.5500 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.0000 & 0.0000 & 0.0000 \\
0.2000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2500 & 0.1250 & 0.0625 & 0.0000 & 0.5000 & 0.2500 & 0.1250 \\
0.0000 & 0.5500 & 0.2750 & 0.1375 & 0.0000 & 0.5000 & 0.2500 & 0.1250 \\
0.0000 & 0.2000 & 0.1000 & 0.0500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.1250 & 0.1458 & 0.0000 & 0.0000 & 0.2500 & 0.2917 \\
0.0000 & 0.0000 & 0.2750 & 0.3208 & 0.0000 & 0.0000 & 0.2500 & 0.2917 \\
0.0000 & 0.0000 & 0.1000 & 0.1167 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0833 \\
0.0000 & 0.0000 & 0.0000 & 0.0917 & 0.0000 & 0.0000 & 0.0000 & 0.0833 \\
0.0000 & 0.0000 & 0.0000 & 0.0333 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right.$$

$$\left.\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\
1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.5000 & 0.2500 \\
0.0000 & 1.0000 & 0.5000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5000 & 0.5833 \\
0.0000 & 0.0000 & 0.5000 & 0.5833 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1667 \\
0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right]$$

$$\mathbf{C}^{10} = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.0000 & 0.0000 & 0.0000 \\ 0.1111 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.5556 & 0.5000 & 0.2500 & 0.1250 \\ 0.5556 & 0.5000 & 0.2500 & 0.1250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.5000 & 0.7500 & 0.7500 \\ 0.3333 & 0.5000 & 0.7500 & 0.7500 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1250 \\ 0.0000 & 0.0000 & 0.0000 & 0.1250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

$$\begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0370 & 0.0000 & 0.0000 & 0.0000 \\ 0.0741 & 0.0000 & 0.0000 & 0.0000 & 0.0617 & 0.0000 & 0.0000 & 0.0000 \\ 0.0370 & 0.0000 & 0.0000 & 0.0000 & 0.0123 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1852 & 0.1667 & 0.0833 & 0.0417 \\ 0.3704 & 0.3333 & 0.1667 & 0.0833 & 0.3086 & 0.2778 & 0.1389 & 0.0694 \\ 0.1852 & 0.1667 & 0.0833 & 0.0417 & 0.0617 & 0.0556 & 0.0278 & 0.0139 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.1667 & 0.2500 & 0.2500 \\ 0.2222 & 0.3333 & 0.5000 & 0.5000 & 0.1852 & 0.2778 & 0.4167 & 0.4167 \\ 0.1111 & 0.1667 & 0.2500 & 0.2500 & 0.0370 & 0.0556 & 0.0833 & 0.0833 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\ 0.0000 & 0.0000 & 0.0000 & 0.0833 & 0.0000 & 0.0000 & 0.0000 & 0.0694 \\ 0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0139 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0035 \end{bmatrix}$$

$$\mathbf{C}^{11} = \left[\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0021 & 0.0000 & 0.0000 & 0.0000 & 0.0063 & 0.0000 & 0.0000 & 0.0000 \\
0.0188 & 0.0000 & 0.0000 & 0.0000 & 0.0250 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\hline
0.0312 & 0.0250 & 0.0167 & 0.0111 & 0.0937 & 0.0750 & 0.0500 & 0.0333 \\
0.2812 & 0.2250 & 0.1500 & 0.1000 & 0.3750 & 0.3000 & 0.2000 & 0.1333 \\
0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.2500 & 0.1667 & 0.1111 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\hline
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0319 & 0.0389 & 0.0444 & 0.0444 & 0.0958 & 0.1167 & 0.1333 & 0.1333 \\
0.2875 & 0.3500 & 0.4000 & 0.4000 & 0.3833 & 0.4667 & 0.5333 & 0.5333 \\
0.0417 & 0.0833 & 0.1667 & 0.2000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\hline
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0035 & 0.0069 & 0.0139 & 0.0278 & 0.0069 & 0.0139 & 0.0278 & 0.0556 \\
0.0139 & 0.0278 & 0.0556 & 0.1111 & 0.0139 & 0.0278 & 0.0556 & 0.1111 \\
0.0000 & 0.0000 & 0.0000 & 0.0111 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right.$$

$$\left.\begin{array}{cccccccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0078 & 0.0000 & 0.0000 & 0.0000 \\
0.0187 & 0.0000 & 0.0000 & 0.0000 & 0.0172 & 0.0000 & 0.0000 & 0.0000 \\
0.0125 & 0.0000 & 0.0000 & 0.0000 & 0.0063 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1172 & 0.0937 & 0.0625 & 0.0417 \\
\hline
0.2812 & 0.2250 & 0.1500 & 0.1000 & 0.2578 & 0.2063 & 0.1375 & 0.0917 \\
0.1875 & 0.1500 & 0.1000 & 0.0667 & 0.0938 & 0.0750 & 0.0500 & 0.0333 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\hline
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1198 & 0.1458 & 0.1667 & 0.1667 \\
0.2875 & 0.3500 & 0.4000 & 0.4000 & 0.2635 & 0.3208 & 0.3667 & 0.3667 \\
0.1917 & 0.2333 & 0.2667 & 0.2667 & 0.0958 & 0.1167 & 0.1333 & 0.1333 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\hline
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0052 & 0.0104 & 0.0208 & 0.0417 \\
0.0139 & 0.0278 & 0.0556 & 0.1111 & 0.0122 & 0.0243 & 0.0486 & 0.0972 \\
0.0069 & 0.0139 & 0.0278 & 0.0556 & 0.0035 & 0.0069 & 0.0139 & 0.0278 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{array}\right]$$

$$\mathbf{C}^{17} = \begin{bmatrix}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0208 & 0.0000 & 0.0000 & 0.0000 & 0.0417 & 0.0000 & 0.0000 & 0.0000 \\
0.2500 & 0.0000 & 0.0000 & 0.0000 & 0.2500 & 0.0000 & 0.0000 & 0.0000 \\
0.0417 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0833 & 0.0833 & 0.0417 & 0.0208 & 0.1667 & 0.1667 & 0.0833 & 0.0417 \\
0.4500 & 0.6000 & 0.3000 & 0.1500 & 0.4500 & 0.6000 & 0.3000 & 0.1500 \\
0.0750 & 0.1000 & 0.0500 & 0.0250 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0035 & 0.0069 & 0.0139 & 0.0122 & 0.0069 & 0.0139 & 0.0278 & 0.0243 \\
0.0799 & 0.1597 & 0.3194 & 0.2795 & 0.0972 & 0.1944 & 0.3889 & 0.3403 \\
0.0312 & 0.0937 & 0.2812 & 0.2578 & 0.0250 & 0.0750 & 0.2250 & 0.2062 \\
0.0021 & 0.0062 & 0.0187 & 0.0172 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0052 & 0.0000 & 0.0000 & 0.0000 & 0.0104 \\
0.0000 & 0.0000 & 0.0000 & 0.1198 & 0.0000 & 0.0000 & 0.0000 & 0.1458 \\
0.0000 & 0.0000 & 0.0000 & 0.1172 & 0.0000 & 0.0000 & 0.0000 & 0.0937 \\
0.0000 & 0.0000 & 0.0000 & 0.0078 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{bmatrix}$$

$$\begin{bmatrix}
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0111 & 0.0000 & 0.0000 & 0.0000 \\
0.0833 & 0.0000 & 0.0000 & 0.0000 & 0.1000 & 0.0000 & 0.0000 & 0.0000 \\
0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1111 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0444 & 0.0444 & 0.0222 & 0.0111 \\
0.3333 & 0.3333 & 0.1667 & 0.0833 & 0.4000 & 0.4000 & 0.2000 & 0.1000 \\
0.3000 & 0.4000 & 0.2000 & 0.1000 & 0.2000 & 0.2667 & 0.1333 & 0.0667 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0139 & 0.0278 & 0.0556 & 0.0486 & 0.0278 & 0.0556 & 0.1111 & 0.0972 \\
0.1111 & 0.2222 & 0.4444 & 0.3889 & 0.1111 & 0.2222 & 0.4444 & 0.3889 \\
0.0167 & 0.0500 & 0.1500 & 0.1375 & 0.0111 & 0.0333 & 0.1000 & 0.0917 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0208 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\
0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 \\
0.0000 & 0.0000 & 0.0000 & 0.0625 & 0.0000 & 0.0000 & 0.0000 & 0.0417 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{bmatrix}$$

Appendix B

Basis function derivatives

We now derive the derivatives of a rational T-spline basis function with respect to the physical coordinates. In the two-dimensional case, we can compute the first-order derivatives by differentiation of

$$R(\xi,\eta) = \tilde{R}(x(\xi,\eta), y(\xi,\eta)) \tag{B.1}$$

to yield the system

$$\begin{pmatrix} R_\xi \\ R_\eta \end{pmatrix} = \begin{bmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{bmatrix} \begin{pmatrix} \tilde{R}_x \\ \tilde{R}_y \end{pmatrix} \tag{B.2}$$

where the subscripts are used to indicate differentiation. Since efficient and robust algorithms exist for the computation of the derivatives with respect to the parametric coordinates, e.g. [88], the basis function derivatives with respect to the physical coordinates are obtained by

$$\begin{pmatrix} \tilde{R}_x \\ \tilde{R}_y \end{pmatrix} = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \begin{bmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{bmatrix} \begin{pmatrix} R_\xi \\ R_\eta \end{pmatrix} \tag{B.3}$$

Using these results, the second-order basis function derivatives with respect to the physical coordinate are obtained by solving the system

$$\begin{bmatrix} x_\xi^2 & 2x_\xi y_\xi & y_\xi^2 \\ x_\xi x_\eta & x_\xi y_\eta + x_\eta y_\xi & y_\xi y_\eta \\ x_\eta^2 & 2x_\eta y_\eta & y_\eta^2 \end{bmatrix} \begin{pmatrix} \tilde{R}_{xx} \\ \tilde{R}_{xy} \\ \tilde{R}_{yy} \end{pmatrix} = \begin{pmatrix} R_{\xi\xi} \\ R_{\xi\eta} \\ R_{\eta\eta} \end{pmatrix} - \begin{bmatrix} x_{\xi\xi} & y_{\xi\xi} \\ x_{\xi\eta} & y_{\xi\eta} \\ x_{\eta\eta} & y_{\eta\eta} \end{bmatrix} \begin{pmatrix} \tilde{R}_x \\ \tilde{R}_y \end{pmatrix} \tag{B.4}$$

and subsequently, the third-order derivatives are obtained as

$$\begin{bmatrix} x_\xi^3 & 3x_\xi^2 y_\xi & 3x_\xi y_\xi^2 & y_\xi^3 \\ x_\xi^2 x_\eta & x_\xi^2 y_\eta + 2x_\xi x_\eta y_\xi & x_\eta y_\xi^2 + 2x_\xi y_\xi y_\eta & y_\xi^2 y_\eta \\ x_\xi x_\eta^2 & x_\eta^2 y_\xi + 2x_\xi x_\eta y_\eta & x_\xi y_\eta^2 + 2x_\eta y_\xi y_\eta & y_\xi y_\eta^2 \\ x_\eta^3 & 3x_\eta^2 y_\eta & 3x_\eta y_\eta^2 & y_\eta^3 \end{bmatrix} \begin{pmatrix} \tilde{R}_{xxx} \\ \tilde{R}_{xxy} \\ \tilde{R}_{xyy} \\ \tilde{R}_{yyy} \end{pmatrix} = \begin{pmatrix} R_{\xi\xi\xi} \\ R_{\xi\xi\eta} \\ R_{\xi\eta\eta} \\ R_{\eta\eta\eta} \end{pmatrix}$$

$$- \begin{bmatrix} 3x_\xi x_{\xi\xi} & 3x_\xi y_{\xi\xi} + 3x_{\xi\xi} y_\xi & 3y_\xi y_{\xi\xi} \\ 2x_\xi x_{\xi\eta} + x_\eta x_{\xi\xi} & 2x_{\xi\eta} y_\xi + 2x_\xi y_{\xi\eta} + x_{\xi\xi} y_\eta + x_\eta y_{\xi\xi} & 2y_\xi y_{\xi\eta} + y_\eta y_{\xi\xi} \\ 2x_\eta x_{\xi\eta} + x_\xi x_{\eta\eta} & 2x_{\xi\eta} y_\eta + 2x_\eta y_{\xi\eta} + x_{\eta\eta} y_\xi + x_\xi y_{\eta\eta} & 2y_\eta y_{\xi\eta} + y_\xi y_{\eta\eta} \\ 3x_\eta x_{\eta\eta} & 3x_\eta y_{\eta\eta} + 3x_{\eta\eta} y_\eta & 3y_\eta y_{\eta\eta} \end{bmatrix} \begin{pmatrix} \tilde{R}_{xx} \\ \tilde{R}_{xy} \\ \tilde{R}_{yy} \end{pmatrix}$$

$$- \begin{bmatrix} x_{\xi\xi\xi} & y_{\xi\xi\xi} \\ x_{\xi\xi\eta} & y_{\xi\xi\eta} \\ x_{\xi\eta\eta} & y_{\xi\eta\eta} \\ x_{\eta\eta\eta} & y_{\eta\eta\eta} \end{bmatrix} \begin{pmatrix} \tilde{R}_x \\ \tilde{R}_y \end{pmatrix} \tag{B.5}$$

Similar results can be obtained in the three-dimensional case.

Appendix C

T-NURCC subdivision

We develop an element form for the elements near extraordinary points using T-NURCC subdivision and demonstrate several of the difficulties incorporating the *infinite* piecewise polynomial representation of extraordinary elements into a finite element framework.

In a T-NURCC [106], the extraordinary elements which correspond to the one-ring neighbhorhood (see Chapter 6 for the underlying topological concepts) are defined through a subdivision procedure. As shown in Figure C.1, during the subdivision process the position of two-ring control points are updated and new control points are introduced. These new control points are written as convex combinations of existing control points. The subdivision rules for T-NURCCs are presented in Appendix C.5. We note that these rules were derived assuming that each iteration of subdivision splits existing knot intervals in half. A distinguishing feature of T-NURCC subdivision is that the subdivision process remains localized to the two-ring neighborhood of an extraordinary point.

C.1 The element subdivision matrix

T-NURCC subdivision is applied to an extraordinary element through the action of an *element subdivision matrix*. We denote the control points supported by element e as $\mathbf{P}_0^{e\,T} = (\mathbf{P}_{0,1}^e, \ldots, \mathbf{P}_{0,K}^e)^T$. A single iteration of T-NURCC subdivision

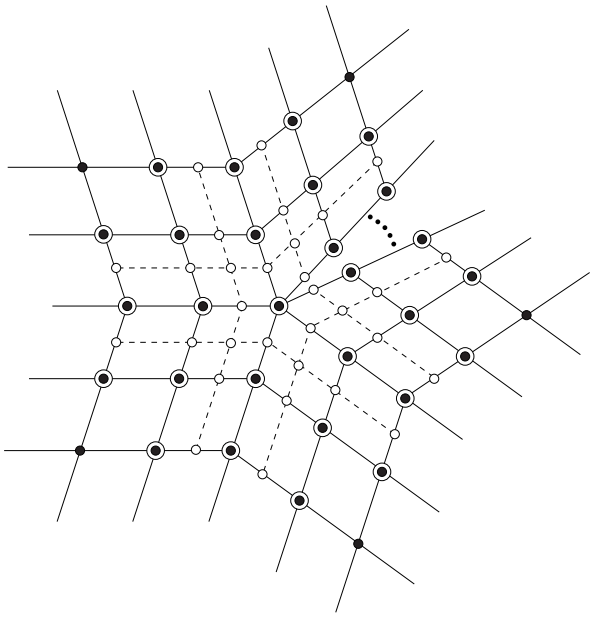


Figure C.1: A T-mesh after one T-NURCC subdivision step. The position of the vertices represented by hollow circles is computed as a convex combination of existing control points (solid circles.) The subdivision process remains localized to the T-mesh elements in the two-ring neighborhood of the extraordinary point.

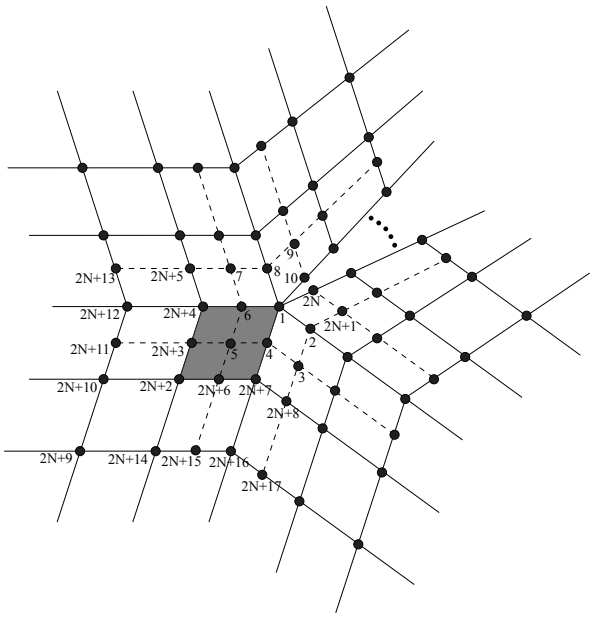


Figure C.2: An extraordinary element (shaded element) after one application of T-NURCC subdivision.

generates $M = K+9$ new control points as shown in Figure C.2. These new control points are convex combinations of the original K control points. The inner K new control points are written as linear combinations of original control points. In other words,

$$\mathbf{P}_1^e = \mathbf{S}_1^e \mathbf{P}_0^e \tag{C.1}$$

where $\mathbf{S}_1^e$ is a $K \times K$ T-spline element subdivision matrix and $\mathbf{P}_1^{e\,T} = (\mathbf{P}_{1,1}^e, \ldots, \mathbf{P}_{1,K}^e)^T$. Additionally, the entire set of M new control points can be defined using an extended element subdivision matrix $\bar{\mathbf{S}}_1^e$ of size $M \times K$ such that

$$\bar{\mathbf{P}}_1^e = \bar{\mathbf{S}}_1^e \mathbf{P}_0^e \tag{C.2}$$

where $\bar{\mathbf{P}}_1^{e\ T} = (\mathbf{P}_{1,1}^e, \ldots, \mathbf{P}_{1,M}^e)^T$. Subdivision can be applied repeatedly to create an infinite sequence of control vertices. In other words,

$$\mathbf{P}_n^e = \begin{cases} \mathbf{S}_1^e \mathbf{P}_0^e, & n = 1 \\ \mathbf{S}_2^e \mathbf{S}_1^e \mathbf{P}_0^e, & n = 2 \\ (\mathbf{S}_3^e)^{n-2} \mathbf{S}_2^e \mathbf{S}_1^e \mathbf{P}_0^e, & n \geq 3 \end{cases} \tag{C.3}$$

and

$$\bar{\mathbf{P}}_n^e = \begin{cases} \bar{\mathbf{S}}_1^e \mathbf{P}_0^e, & n = 1 \\ \bar{\mathbf{S}}_2^e \mathbf{S}_1^e \mathbf{P}_0^e, & n = 2\ . \\ \bar{\mathbf{S}}_3^e (\mathbf{S}_3^e)^{n-3} \mathbf{S}_2^e \mathbf{S}_1^e \mathbf{P}_0^e, & n \geq 3 \end{cases} \tag{C.4}$$

T-NURCC subdivision produces three distinct element subdivision matrices. For $n \geq 3$, $\mathbf{S}_n^e = \mathbf{S}_3^e$. In all cases, the linear combinations presented in Appendix C.5 are used to define the element subdivision matrices.

C.2 Element definition

To define an extraordinary element, we partition a domain $\hat{\Omega} = [-1, 1] \otimes [-1, 1]$ into an infinite set of subdomains $\{\hat{\Omega}_{k,n}\}$, $n \geq 1, k = 1, 2, 3$, as shown in Figure C.3. Each subdomain with index n is four times smaller than subdomains with index $n - 1$. In other words,

$$\hat{\Omega}_{1,n} = \left[\frac{1 - 2^{n-1}}{2^{n-1}}, \frac{1 - 2^{n-2}}{2^{n-2}}\right] \otimes \left[-1, \frac{1 - 2^{n-1}}{2^{n-1}}\right] \tag{C.5}$$

$$\hat{\Omega}_{2,n} = \left[\frac{1 - 2^{n-1}}{2^{n-1}}, \frac{1 - 2^{n-2}}{2^{n-2}}\right] \otimes \left[\frac{1 - 2^{n-1}}{2^{n-1}}, \frac{1 - 2^{n-2}}{2^{n-2}}\right] \tag{C.6}$$

$$\hat{\Omega}_{3,n} = \left[-1, \frac{1 - 2^{n-1}}{2^{n-1}}\right] \otimes \left[\frac{1 - 2^{n-1}}{2^{n-1}}, \frac{1 - 2^{n-2}}{2^{n-2}}\right]. \tag{C.7}$$

An extraordinary element, $\mathbf{x}^e(\hat{\boldsymbol{\xi}}) : \hat{\Omega} \to \Omega^e$, is then constructed by defining its restriction to each subdomain, $\hat{\Omega}_{k,n}$, to be equal to a polynomial subelement

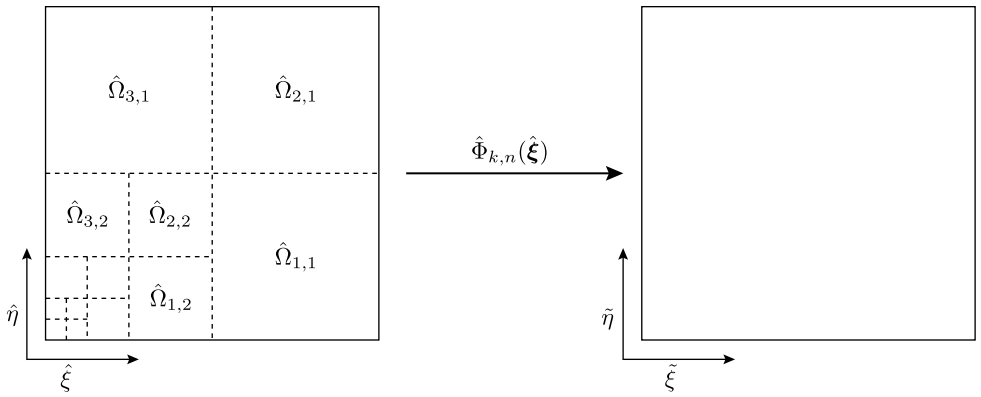


Figure C.3: An extraordinary element defined through T-NURCC subdivision.

$\mathbf{x}^e_{k,n}(\hat{\boldsymbol{\xi}}) : \hat{\Omega}_{k,n} \to \Omega_{k,n}$. In other words,

$$\mathbf{x}^e(\hat{\boldsymbol{\xi}})|_{\hat{\Omega}_{k,n}} = \mathbf{x}^e_{k,n}(\hat{\boldsymbol{\xi}}). \tag{C.8}$$

The polynomial subelement, $\mathbf{x}^e_{k,n}(\hat{\boldsymbol{\xi}})$, is constructed by selecting from $\bar{\mathbf{P}}^e_n$ the control points which define a tensor product bicubic non-uniform B-spline over $\hat{\Omega}_{k,n}$. The sixteen control points, $\mathbf{P}^e_{k,n}$, are selected from $\bar{\mathbf{P}}^e_n$ by defining an appropriate boolean matrix, $\mathbf{B}_k$, of size $16 \times M$ (see [112] for the exact form of $\mathbf{B}_k$) such that

$$\mathbf{P}^e_{k,n} = \mathbf{B}_k \bar{\mathbf{P}}^e_n. \tag{C.9}$$

The subelement, $\mathbf{x}^e_{k,n}(\hat{\boldsymbol{\xi}}) : \hat{\Omega}_{k,n} \to \Omega^e_{k,n}$, is then defined through Bézier extraction as

$$\mathbf{x}^e_{k,n}(\hat{\boldsymbol{\xi}}) = (\mathbf{P}^e_{k,n})^T \mathbf{N}^e_{k,n}(\hat{\boldsymbol{\xi}}) \tag{C.10}$$

$$= (\mathbf{P}^e_{k,n})^T \mathbf{N}^e_{k,n}(\boldsymbol{\Phi}_{k,n}(\hat{\boldsymbol{\xi}})) \tag{C.11}$$

$$= (\mathbf{P}^e_{k,n})^T \mathbf{C}^e_{k,n} \mathbf{B}(\boldsymbol{\Phi}_{k,n}(\hat{\boldsymbol{\xi}})) \tag{C.12}$$

$$= (\mathbf{P}^e_{k,n})^T \mathbf{C}^e_{k,n} \mathbf{B}(\tilde{\boldsymbol{\xi}}), \tag{C.13}$$

where $\mathbf{N}^e_{k,n}(\hat{\boldsymbol{\xi}})^T = (N^{e,1}_{k,n}(\hat{\boldsymbol{\xi}}), \ldots, N^{e,16}_{k,n}(\hat{\boldsymbol{\xi}}))^T$ is the vector of non-uniform B-spline basis functions defined over subelement $\hat{\Omega}_{k,n}$, $\mathbf{B}(\tilde{\boldsymbol{\xi}})^T = (B_1(\tilde{\boldsymbol{\xi}}), \ldots, B_{16}(\tilde{\boldsymbol{\xi}}))^T$ is the vector of Bernstein basis functions defined over the parent element $\tilde{\Omega}$, and $\mathbf{C}^e_{k,n}$ is the element extraction operator corresponding to subelement $\hat{\Omega}_{k,n}$. The map $\boldsymbol{\Phi}_{k,n} : \hat{\Omega}_{k,n} \to \tilde{\Omega}$ (shown in Figure C.3) is defined as

$$\hat{\boldsymbol{\Phi}}_{k,n}(\hat{\xi}, \hat{\eta}) = \begin{cases} (2^n(\hat{\xi}+1) - 3, 2^n(\hat{\eta}+1) - 1), & k = 1 \\ (2^n(\hat{\xi}+1) - 3, 2^n(\hat{\eta}+1) - 3), & k = 2 \\ (2^n(\hat{\xi}+1) - 1, 2^n(\hat{\eta}+1) - 3), & k = 3. \end{cases} \tag{C.14}$$

Notice that Bézier extraction allows us to standardize the form of the element basis on the parent domain, $\tilde{\Omega}$. In other words, each subelement is defined in terms of the same set of Bernstein basis function regardless of e, k, or n. For each extraordinary element there are nine unique subelement extraction operators, $\mathbf{C}^e_{k,n}$, $n = 1, 2, 3$, $k = 1, 2, 3$. For $n \geq 3$, $\mathbf{C}^e_{k,n} = \mathbf{C}^e_{k,3}$.

Using (C.4), (C.9), and (C.12) we can write the extraordinary element as

$$\mathbf{x}^e(\hat{\boldsymbol{\xi}}) = \begin{cases} (\mathbf{P}^e_0)^T \left[(\mathbf{B}_k \bar{\mathbf{S}}^e_1)^T \mathbf{C}^e_{k,1}\right] \mathbf{B}(\hat{\boldsymbol{\Phi}}_{k,1}(\hat{\boldsymbol{\xi}})), & n = 1, k = 1, 2, 3 \\ (\mathbf{P}^e_0)^T \left[(\mathbf{B}_k \bar{\mathbf{S}}^e_2 \mathbf{S}^e_1)^T \mathbf{C}^e_{k,2}\right] \mathbf{B}(\hat{\boldsymbol{\Phi}}_{k,2}(\hat{\boldsymbol{\xi}})), & n = 2, k = 1, 2, 3 \\ (\mathbf{P}^e_0)^T \left[(\mathbf{B}_k \bar{\mathbf{S}}^e_3 (\mathbf{S}^e_3)^{n-3} \mathbf{S}^e_2 \mathbf{S}^e_1)^T \mathbf{C}^e_{k,3}\right] \mathbf{B}(\hat{\boldsymbol{\Phi}}_{k,n}(\hat{\boldsymbol{\xi}})), & n \geq 3, k = 1, 2, 3 \end{cases} \tag{C.15}$$

or more compactly as

$$\mathbf{x}^e(\hat{\boldsymbol{\xi}}) = (\mathbf{P}^e_0)^T \hat{\mathbf{C}}^e_{k,n} \mathbf{B}(\hat{\boldsymbol{\Phi}}_{k,n}(\hat{\boldsymbol{\xi}})) \tag{C.16}$$

where $\hat{\mathbf{C}}^e_{k,n}$ is a *subdivision element extraction operator*. Notice that due to the piecewise polynomial representation of the extraordinary element the subdivision extraction operator changes from point to point throughout the element. This greatly complicates its use in finite element analysis when compared to the extraordinary element technology presented in Chapter 6.

C.3 Exploiting eigenstructure

To evaluate an extraordinary element at a location close to the extraordinary point is prohibitively expensive. In other words, evaluating subelement, $\mathbf{x}_{k,n}^e(\hat{\boldsymbol{\xi}})$, requires n subdivision matrix multiplies. To overcome this limitation, the eigenstructure of $\mathbf{S}_3^e$ can be exploited. Consider an eigenvalue decomposition of $\mathbf{S}_3^e$ such that

$$\mathbf{S}_3^e = \mathbf{V}^e \boldsymbol{\Lambda}^e (\mathbf{V}^e)^{-1} \tag{C.17}$$

where $\boldsymbol{\Lambda}^e$ is a diagonal matrix containing the eigenvalues of $\mathbf{S}_3^e$ and $\mathbf{V}^e$ is an invertible matrix whose columns are the eigenvectors. For T-NURCC subdivision, given an eigenvalue, λ_a, $a = 1, \ldots, K$, we have that $0 \leq \lambda_a \leq 1$. Additionally, for any valence N, the largest eigenvalue is exactly 1.

Using the subdivision extraction operator, $\hat{\mathbf{C}}_{k,n}^e$, for $n \geq 3$ and (C.17) we have that

$$\begin{aligned}
\hat{\mathbf{C}}_{k,n}^e &= (\mathbf{B}_k \bar{\mathbf{S}}_3^e (\mathbf{S}_3^e)^{n-3} \mathbf{S}_2^e \mathbf{S}_1^e)^T \mathbf{C}_{k,3}^e && \text{(C.18)} \\
&= (\mathbf{B}_k \bar{\mathbf{S}}_3^e \mathbf{V}^e (\boldsymbol{\Lambda}^e)^{n-3} (\mathbf{V}^e)^{-1} \mathbf{S}_2^e \mathbf{S}_1^e)^T \mathbf{C}_{k,3}^e && \text{(C.19)} \\
&= \left[(\mathbf{V}^e)^{-1} \mathbf{S}_2^e \mathbf{S}_1^e\right]^T (\boldsymbol{\Lambda}^e)^{n-3} \left[(\mathbf{B}_k \bar{\mathbf{S}}_3^e)^T \mathbf{C}_{k,3}^e\right] && \text{(C.20)} \\
&= \mathbf{L}^e (\boldsymbol{\Lambda}^e)^{n-3} \mathbf{R}_k^e. && \text{(C.21)}
\end{aligned}$$

Notice that n is only used in the scaling of the eigenvalue matrix $\boldsymbol{\Lambda}^e$. Both $\mathbf{L}^e$ and $\mathbf{R}_k^e$ are independent of n and only $\mathbf{R}_k^e$ depends on k. Using (C.15) and (C.21) we have that

$$\hat{\mathbf{C}}_{k,n}^e = \begin{cases} (\mathbf{B}_k \bar{\mathbf{S}}_1^e)^T \mathbf{C}_{k,1}^e, & n = 1 \\ (\mathbf{B}_k \bar{\mathbf{S}}_2^e \mathbf{S}_1^e)^T \mathbf{C}_{k,2}^e, & n = 2 \\ \mathbf{L}^e (\boldsymbol{\Lambda}^e)^{n-3} \mathbf{R}_k^e, & n \geq 3 \end{cases}. \tag{C.22}$$

C.4 The challenge of numerical integration

Extraordinary elements defined through T-NURCC subdivision have a piecewise polynomial representation. As such, Gauss quadrature *cannot* be applied directly. To demonstrate, patch tests, as described in Section 6.4, were attempted using T-NURCC subdivision. The T-mesh in Figure 6.8 was used with uniform knot intervals. As expected, extraordinary elements defined through T-NURCC subdivision *do not* pass patch tests due to numerical integration errors.

To quantify the errors, the L^2 and H^1 norms of the error, for the stretch patch test in the x direction described in Figure 6.7, are plotted in Figure C.4. In this case, the gauss rule is increased over each one-ring extraordinary element and held fixed at four-point quadrature for all non-extraordinary elements. As the gauss rule is increased the error decreases slowly but never disappears.

Instead of applying gauss quadrature to each extraordinary element we can integrate each *subelement* on subdivision level n using gauss quadrature. Gauss quadrature exactly integrates the B-splines defining a subelement (see Section C.2). As $n \to \infty$, the global integration errors converge to zero. Using this approach, the L^2 and H^1 norms of the error are plotted in Figure C.5. In this example, four-point gauss quadrature is used for non-extraordinary elements and all extraordinary subelements.

It is obvious that integrating each subelement using gauss quadrature is not a feasible approach due to the number of integration points required. For example, to integrate an extraordinary element up to subdivision level $n = 25$, using a four-point gauss rule, requires $25(3)(16) = 1200$ points. To overcome this limitation, new quadrature rules must be developed which take into account the piecewise polynomial representation of an extraordinary element defined through T-NURCC subdivision.

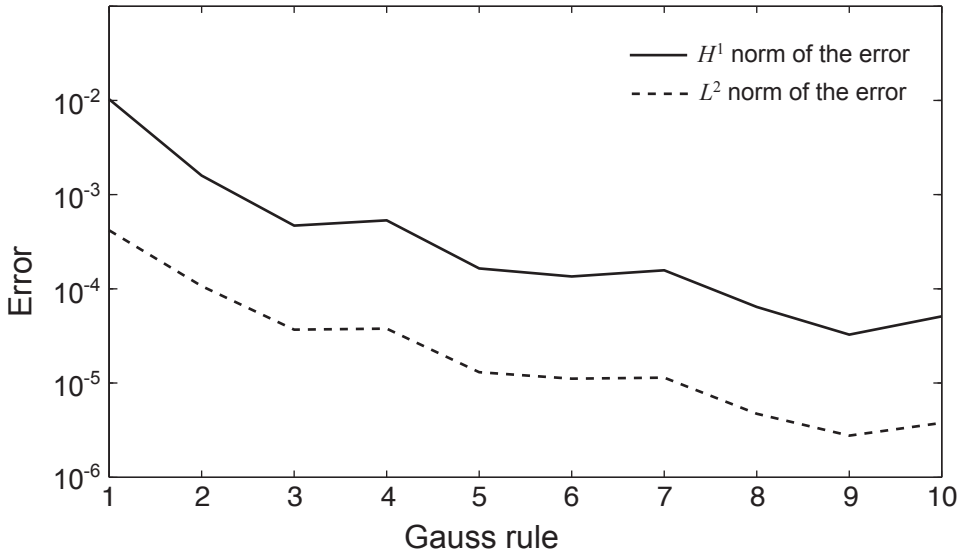


Figure C.4: The L^2 and H^1 norms of the error for the stretch patch test described in Figure 6.7 for increasing gauss rule over each one-ring extraordinary element. All non-extraordinary elements are integrated using four-point gauss quadrature.

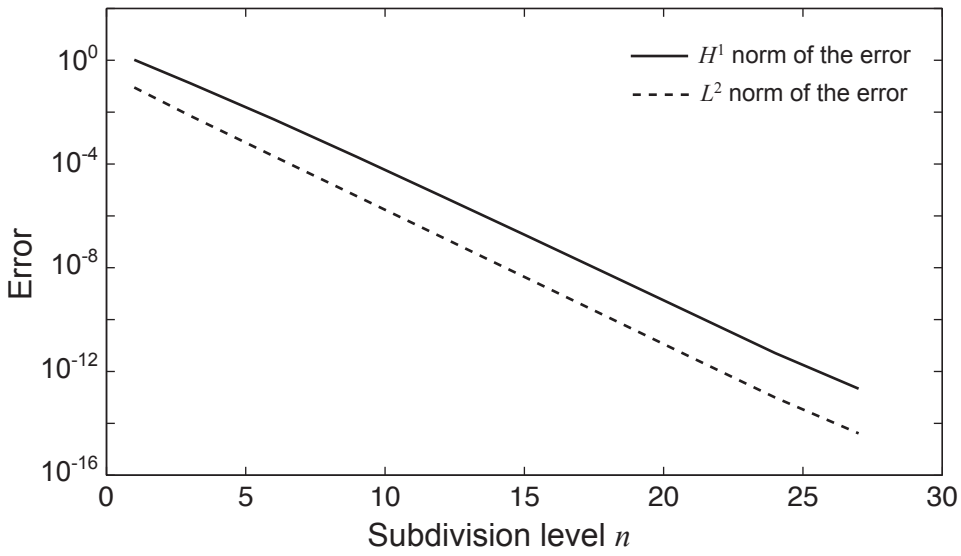


Figure C.5: The L^2 and H^1 norms of the error for the stretch patch test described in Figure 6.7 for exact integration of each subelement on subdivision level n. Four-point gauss quadrature is used in all cases.

C.5 Subdivision rules

We present the subdivision rules for T-NURCCs. Figure C.6 establishes the notation used in describing the subdivision scheme. Due to symmetry only one quadrant of a two-ring neighborhood is shown. The vertices labeled $\mathbf{P}^i_{jk}$ denote initial control points before subdivision is performed. The vertices labeled $\mathbf{Q}^i_{jk}$ denote control points resulting from subdivision. The knot intervals are labeled a^i_j. The dashed lines represent edges added during a subdivision step. We then have that

$$\mathbf{Q}^i_{11} = \frac{w_1\mathbf{P}^i_{00} + w_2\mathbf{P}^i_{01} + w_3\mathbf{P}^i_{10} + w_4\mathbf{P}^i_{11}}{w_1 + w_2 + w_3 + w_4} \tag{C.23}$$

where

$$w_1 = (4a^i_1 + 2a^i_0)(4a^{i+1}_1 + 2a^{i+1}_0) \tag{C.24}$$

$$w_2 = (4a^i_1 + 2a^i_0)(2a^{i+1}_0 + 3a^{i-1}_0 + a^{i+3}_0) \tag{C.25}$$

$$w_3 = (2a^i_0 + 3a^{i+2}_0 + a^{i-2}_0)(4a^{i+1}_1 + 2a^{i+1}_0) \tag{C.26}$$

$$w_4 = (2a^i_0 + 3a^{i+2}_0 + a^{i-2}_0)(2a^{i+1}_0 + 3a^{i-1}_0 + a^{i+3}_0). \tag{C.27}$$

$$\mathbf{Q}^i_{10} = \mathbf{Q}^{i-1}_{01} = \frac{1}{2}\mathbf{M}^i + \frac{a^{i-1}_0\mathbf{Q}^i_{11} + a^{i+1}_0\mathbf{Q}^{i-1}_{11}}{2(a^{i+1}_0 + a^{i-1}_0)} \tag{C.28}$$

where

$$\mathbf{M}^i = \frac{(a^i_0 + 2a^i_1)\mathbf{P}_{00} + (a^{i+2}_0 + a^{i-2}_0 + a^i_0)\mathbf{P}^i_{10}}{a^{i+2}_0 + a^{i-2}_0 + 2a^i_0 + 2a^i_1}. \tag{C.29}$$

$$\mathbf{Q}^i_{21} = \frac{w_1\mathbf{P}^i_{10} + w_2\mathbf{P}^i_{11} + w_3\mathbf{P}^i_{20} + w_4\mathbf{P}^i_{21}}{w_1 + w_2 + w_3 + w_4}. \tag{C.30}$$

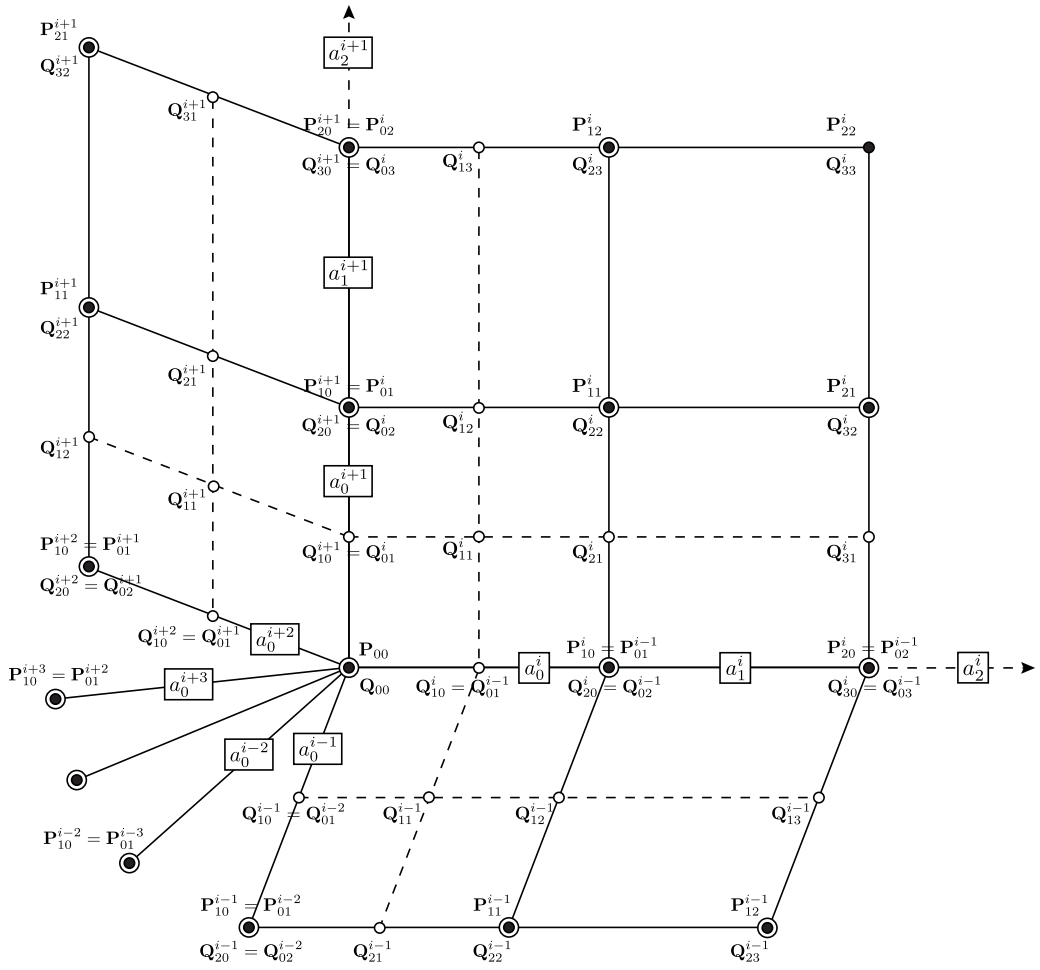


Figure C.6: Notational conventions used to describe T-NURCC subdivision. Due to symmetry only one quadrant of a two-ring neighborhood is shown. The vertices labeled $\mathbf{P}_{jk}^{i}$ denote initial control points before subdivision is performed. The vertices labeled $\mathbf{Q}_{jk}^{i}$ denote control points resulting from subdivision. The knot intervals are labeled a_j^i. The dashed lines represent edges added during a subdivision step.

where

$$w_1 = (a_0^i + 2a_1^i + 2a_2^i)(a_0^{i+1} + 2a_1^{i+1}) \tag{C.31}$$

$$w_2 = (a_0^i + 2a_1^i + 2a_2^i)(2a_0^{i-1} + a_0^{i+1}) \tag{C.32}$$

$$w_3 = a_0^i(2a_1^{i+1} + a_0^{i+1}) \tag{C.33}$$

$$w_4 = a_0^i(a_0^{i+1} + 2a_0^{i-1}) \tag{C.34}$$

$$\mathbf{Q}_{12}^i = \frac{w_1\mathbf{P}_{01}^i + w_2\mathbf{P}_{11}^i + w_3\mathbf{P}_{02}^i + w_4\mathbf{P}_{12}^i}{w_1 + w_2 + w_3 + w_4}. \tag{C.35}$$

where

$$w_1 = (a_0^{i+1} + 2a_1^{i+1} + 2a_2^{i+1})(a_0^i + 2a_1^i) \tag{C.36}$$

$$w_2 = (a_0^{i+1} + 2a_1^{i+1} + 2a_2^{i+1})(2a_0^{i+2} + a_0^i) \tag{C.37}$$

$$w_3 = a_0^{i+1}(2a_1^i + a_0^i) \tag{C.38}$$

$$w_4 = a_0^{i+1}(a_0^i + 2a_0^{i+2}) \tag{C.39}$$

$$\mathbf{Q}_{31}^i = \frac{(a_0^{i+1} + 2a_1^{i+1})\mathbf{P}_{20}^i + (2a_0^{i-1} + a_0^{i+1})\mathbf{P}_{21}^i}{2(a_0^{i-1} + a_0^{i+1} + a_1^{i+1})}. \tag{C.40}$$

$$\mathbf{Q}_{13}^i = \frac{(a_0^i + 2a_1^i)\mathbf{P}_{02}^i + (2a_0^{i+2} + a_0^i)\mathbf{P}_{12}^i}{2(a_0^{i+2} + a_0^i + a_1^i)}. \tag{C.41}$$

$$\mathbf{Q}_{00} = \frac{N-3}{N}\mathbf{P}_{00} + \frac{3}{N}\frac{\sum_{i=1}^N (m^i\mathbf{M}^i + f^i\mathbf{Q}_{11}^i)}{\sum_{i=1}^N (m^i + f^i)} \tag{C.42}$$

where $\mathbf{M}^i$ is defined in (C.29), $\mathbf{Q}_{11}^i$ is defined in (C.23), and

$$m^i = \frac{(a_0^{i-1} + a_0^{i+1})(a_0^{i-2} + a_0^{i+2})}{2} \tag{C.43}$$

$$f^i = a_0^{i-1}a_0^{i+2}. \tag{C.44}$$

$$\mathbf{Q}_{22}^{i} = \frac{w_1\mathbf{P}_{11}^{i} + w_2\mathbf{P}_{12}^{i} + w_3\mathbf{P}_{21}^{i} + w_4\mathbf{P}_{22}^{i}}{w_1 + w_2 + w_3 + w_4} \tag{C.45}$$

where

$$w_1 = (a_0^i + 2a_1^i + 2a_2^i)(a_0^{i+1} + 2a_1^{i+1} + 2a_2^{i+1}) \tag{C.46}$$

$$w_2 = (a_0^i + 2a_1^i + 2a_2^i)a_0^{i+1} \tag{C.47}$$

$$w_3 = a_0^i(a_0^{i+1} + 2a_1^{i+1} + 2a_2^{i+1}) \tag{C.48}$$

$$w_4 = a_0^i a_0^{i+1}. \tag{C.49}$$

$$\mathbf{Q}_{20}^{i} = \mathbf{Q}_{02}^{i-1} = \frac{a_0^{i-1}\mathbf{Q}_{21}^{i} + a_0^{i+1}\mathbf{Q}_{12}^{i-1}}{2(a_0^{i-1} + a_0^{i+1})} + \frac{(a_0^i + 2a_1^i + 2a_2^i)\mathbf{P}_{10}^{i} + a_0^i\mathbf{P}_{20}^{i}}{4(a_0^i + a_1^i + a_2^i)} \tag{C.50}$$

where $\mathbf{Q}_{21}^{i}$ is defined in (C.30) and $\mathbf{Q}_{12}^{i-1}$ is defined in (C.35).

$$\mathbf{Q}_{32}^{i} = \frac{(a_0^{i+1} + 2a_1^{i+1} + 2a_2^{i+1})\mathbf{P}_{21}^{i} + a_0^{i+1}\mathbf{P}_{22}^{i}}{2(a_0^{i+1} + a_1^{i+1} + a_2^{i+1})}. \tag{C.51}$$

$$\mathbf{Q}_{23}^{i} = \frac{(a_0^i + 2a_1^i + 2a_2^i)\mathbf{P}_{12}^{i} + a_0^i\mathbf{P}_{22}^{i}}{2(a_0^i + a_1^i + a_2^i)}. \tag{C.52}$$

$$\mathbf{Q}_{30}^{i} = \mathbf{Q}_{03}^{i-1} = \frac{a_0^{i-1}\mathbf{Q}_{31}^{i} + a_0^{i+1}\mathbf{Q}_{13}^{i-1}}{2(a_0^{i-1} + a_0^{i+1})} + \frac{1}{2}\mathbf{P}_{20}^{i} \tag{C.53}$$

where $\mathbf{Q}_{31}^{i}$ is defined in (C.40) and $\mathbf{Q}_{13}^{i-1}$ is defined in (C.41).

$$\mathbf{Q}_{33}^{i} = \mathbf{P}_{22}^{i}. \tag{C.54}$$

Bibliography

[1] M. Aigner, C. Heinrich, B. Juettler, E. Pilgerstorfer, B. Simeon, and A. V. Vuong. Swept Volume Parameterization for Isogeometric Analysis. In Hancock, E. R. and Martin, R. R. and Sabin, M. A., editor, Mathematics of Surfaces XIII, volume 5654 of Lecture Notes in Computer Science, pages 19–44, 2009.

[2] I. Akkerman, Y. Bazilevs, V. Calo, T. J. R. Hughes, and S. Hulshoff. The role of continuity in residual-based variational multiscale modeling of turbulence. Computational Mechanics, 41:371–378, 2008.

[3] H. Askes, J. Pamin, and R. de Borst. Dispersion analysis and element-free Galerkin solutions of second and fourth-order gradient-enhanced damage models. International Journal for Numerical Methods in Engineering, 49(6):811–832, 2000.

[4] F. Auricchio, L. B. da Veiga, A. Buffa, C. Lovadina, A. Reali, and G. Sangalli. A fully "locking-free" isogeometric approach for plane linear elasticity problems: A stream function formulation. Computer Methods in Applied Mechanics and Engineering, 197:160–172, 2007.

[5] F. Auricchio, L. B. da Veiga, C. Lovadina, and A. Reali. The importance of the exact satisfaction of the incompressibility constraint in nonlinear elasticity: mixed FEMs versus NURBS-based approximations. Computer Methods in Applied Mechanics and Engineering, 199:314–323, 2010.

[6] I. Babuška and J. M. Melenk. The partition of unity method. International Journal for Numerical Methods in Engineering, 40(4):727–758, 1997.

[7] G. P. Bazeley, Y. K. Cheung, B. M. Irons, and O. C. Zienkiewicz. Triangular elements in plate bending—conforming and nonconforming solutions. In Proceedings of the 1st Conference on Matrix Methods in Structural Mechanics, pages 547–576, 1966.

[8] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and residual-based variational multiscale method. Journal of Computational Physics, 229:3402–3414, 2010.

[9] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. Computer Methods in Applied Mechanics and Engineering, 199(5-8):229–263, 2010.

[10] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Computer Methods in Applied Mechanics and Engineering, 197:173–201, 2007.

[11] Y. Bazilevs, V. M. Calo, T. J. R. Hughes, and Y. Zhang. Isogeometric fluid-structure interaction: Theory, algorithms, and computations. Computational Mechanics, 43:3–37, 2008.

[12] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. R. Hughes. Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. Computational Mechanics, 38:310–322, 2006.

[13] Y. Bazilevs, J. R. Gohean, T. J. R. Hughes, R. D. Moser, and Y. Zhang. Patient-specific isogeometric fluid-structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device. Computer Methods in Applied Mechanics and Engineering, 198(45-46):3534–3550, 2009.

[14] Y. Bazilevs and T. J. R. Hughes. NURBS-based isogeometric analysis for the computation of flows about rotating components. Computational Mechanics, 43:143–150, 2008.

[15] Y. Bazilevs, C. Michler, V. M. Calo, and T. J. R. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. Computer Methods in Applied Mechanics and Engineering, 199(13-16):780–790, 2010.

[16] T. Belytschko, H. Chen, J. Xu, and G. Zi. Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment. International Journal for Numerical Methods in Engineering, 58(12):1873–1905, 2003.

[17] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S.-J. Ong. Stress projection for membrane and shear locking in shell finite elements. Computer Methods in Applied Mechanics and Engineering, 51:221–258, 1985.

[18] D. J. Benson, Y. Bazilevs, E. De Luycker, M. C. Hsu, M. A. Scott, T. J. R Hughes, and T. Belytschko. A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to XFEM. International Journal for Numerical Methods in Engineering, 83:765–785, 2010.

[19] D. J. Benson, Y. Bazilevs, M. C. Hsu, and T. J. R. Hughes. Isogeometric shell analysis: The Reissner-Mindlin shell. Computer Methods in Applied

Mechanics and Engineering, 199(5-8):276–289, 2010.

[20] D. J. Benson, Y. Bazilevs, M. C. Hsu, and T. J. R. Hughes. A large deformation, rotation-free, isogeometric shell. International Journal for Numerical Methods in Engineering,, in press, doi:10.1016/j.cma.2010.12.003, 2010.

[21] Å. Björck. Numerical methods for least squares problems. Society for Industrial Mathematics, 1996.

[22] W. Boehm. Visual continuity. Computer-Aided Design, 20(6):307–311, 1988.

[23] M. J. Borden, M. A. Scott, J. A. Evans, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of NURBS. International Journal for Numerical Methods in Engineering,, in press, doi: 10.1002/nme.2968, 2010.

[24] M. J. Borden, M. A. Scott, C. V. Verhoosel, C. M. Landis, and T. J. R. Hughes. Phase-field modeling of dynamic fracture using isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, submitted for publication, 2011.

[25] B. Bourdin, G. A. Francfort, and J. J. Marigo. The variational approach to fracture. Journal of Elasticity, 91(1-3):5–148, April 2008.

[26] A. N. Brooks and T. J. R. Hughes. Streamline upwind / Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering, 32:199–259, 1982.

[27] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. Computer Methods in Applied Mechanics and Engineering, 199(23-24):1437–1445, 2010.

[28] A. Buffa, G. Sangalli, and R. Vazquez. Isogeometric analysis in electromagnetics: B-splines approximation. Computer Methods in Applied Mechanics and Engineering, 199(17-20):1143–1152, 2010.

[29] D. Burkhart, B. Hamann, and G. Umlauf. Isogeometric finite element analysis based on Catmull-Clark subdivision solids. In Computer Graphics Forum, volume 29, pages 1575–1584. Wiley Online Library, 2010.

[30] T. J. Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. ACM Trans. Graph., 28:46:1–46:9, July 2009.

[31] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design, 10:350–355, 1978.

[32] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin shell analysis. International Journal for Numerical Methods in Engineering, 51:813–833, 2001.

[33] F. Cirak, M. Ortiz, and A. Pandolfi. A cohesive approach to thin-shell fracture and fragmentation. Computer Methods in Applied Mechanics and Engineering, 194(21-24):2604–2618, 2005.

[34] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin shell analysis. International Journal for Numerical Methods in Engineering, 47:2039–2072, 2000.

[35] F. Cirak, M. J. Scott, E. K. Antonsson, M. Ortiz, and P. Schröder. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. Computer-Aided Design, 34:137–148, 2002.

[36] E. Cohen, T. Martin, R. M. Kirby, T. Lyche, and R. F. Riesenfeld. Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 199(5-8):334–356, 2010.

[37] P. Costantini, C. Manni, F. Pelosi, and M. L. Sampoli. Quasi-interpolation in isogeometric analysis based on generalized B-splines. Computer Aided Geometric Design, 27(8):656–668, 2010.

[38] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. Isogeometric analysis: Toward Integration of CAD and FEA. Wiley, Chichester, 2009.

[39] J. A. Cottrell, T. J. R. Hughes, and A. Reali. Studies of refinement and continuity in isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 196:4160–4183, 2007.

[40] J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of structural vibrations. Computer Methods in Applied Mechanics and Engineering, 195:5257–5296, 2006.

[41] J. H. P. De Vree, W. A. M. Brekelmans, and M. A. J. Van Gils. Comparison of nonlocal approaches in continuum damage mechanics. Computers and Structures, 55(4):581–588, 1995.

[42] W. L. F. Degen. Explicit continuity conditions for adjacent Bézier surface patches. Computer Aided Geometric Design, 7(1-4):181–189, 1990.

[43] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design, 10(6):356–360, 1978.

[44] M. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. Computer Methods in Applied Mechanics and Engineering, 199(5–8):264–275, 2009.

[45] R. Echter and M. Bischoff. Numerical efficiency, locking and unlocking of NURBS finite elements. Computer Methods in Applied Mechanics and Engineering, 199(5-8):374–382, 2010.

[46] T. Elguedj, Y. Bazilevs, V. M. Calo, and T. J. R. Hughes. $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements. Computer Methods in Applied Mechanics and Engineering, 197:2732–2762, 2008.

[47] J. A. Evans, Y. Bazilevs, I. Babuška, and T. J. R. Hughes. n-widths, sup-infs, and optimality ratios for the k-version of the isogeometric finite element method. Computer Methods in Applied Mechanics and Engineering, 198(21-26):1726–1741, 2009.

[48] G. E. Farin. NURBS Curves and Surfaces: from Projective Geometry to Practical Use. A. K. Peters, Ltd., Natick, MA, 1999.

[49] C. A. Felippa. Course notes for advanced finite element methods. `http://www.colorado.edu/engineering/cas/courses.d/AFEM.d/`, 2011.

[50] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. ACM SIGGRAPH Computer Graphics, 22(4):205–212, 1988.

[51] Z. P. Bazant G. Pijaudier-Cabot. Nonlocal damage theory. Journal of Engineering Mechanics, 113(10):1512–1533, 1987.

[52] M. G. D. Geers, R. De Borst, W. A. M. Brekelmans, and R. H. J. Peerlings. Strain-based transient-gradient damage model for failure analyses. Computer Methods in Applied Mechanics and Engineering, 160(1-2):133–153, 1998.

[53] H. Gomez, V. M. Calo, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of the Cahn-Hilliard phase-field model. Computer Methods in Applied Mechanics and Engineering, 197:4333–4352, 2008.

[54] H. Gomez, T. J. R. Hughes, X. Nogueira, and V. M. Calo. Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. Computer Methods in Applied Mechanics and Engineering, 199(25-28):1828–1840, 2010.

[55] A. Huerta and G. Pijaudier-Cabot. Discretization influence of regularization by two localization limiters. Journal of Engineering Mechanics, 120(6):1198–1218, 1994.

[56] T. J. R. Hughes. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Publications, Mineola, NY, 2000.

[57] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. Computer Methods in Applied Mechanics and Engineering, 194:4135–4195, 2005.

[58] T. J. R. Hughes and L. Franca. A mixed finite element formulation for Reissner-Mindlin plate theory: Uniform convergence of all higher-order spaces.

Computer Methods in Applied Mechanics and Engineering, 67:223–240, 1988.

[59] T. J. R. Hughes, A. Reali, and G. Sangalli. Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: comparison of p-method finite elements with k-method NURBS. Computer Methods in Applied Mechanics and Engineering, 197(49-50):4104–4124, 2008.

[60] T. J. R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 199(5-8):301–313, 2010.

[61] H. Ipson. T-spline merging. Master's thesis, Brigham Young University, April 2005.

[62] J. Kalthoff. Modes of dynamic shear failure in solids. International Journal of Fracture, 101(1):1–31, 2000.

[63] J. F. Kalthoff and S. Winkler. Failure mode transition of high rates of shear loading. In C. Y. Chiem, H. D. Kunze, and L. W. Meyer, editors, Proceedings of the International Conference on Impact Loading and Dynamic Behavior of Materials, volume 1, pages 185–195, 1987.

[64] J. Kiendl, Y. Bazilevs, M. C. Hsu, R. Wuechner, and K. U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff-Love shell structures comprised of multiple patches. Computer Methods in Applied Mechanics and Engineering, 199(37-40):2403–2416, 2010.

[65] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff-Love elements. Computer Methods in Applied Mechanics and Engineering, 198(49-52):3902–3914, 2009.

[66] H. Kim, Y. Seo, and S. Youn. Isogeometric analysis for trimmed CAD surfaces. Computer Methods in Applied Mechanics and Engineering, 198(37-40):2982–2995, 2009.

[67] E. Kuhl. Numerical models for cohesive frictional materials. PhD thesis, University of Stuttgart, 2000.

[68] J. Lemaitre and J. L. Chaboche. Mechanics of solid materials. Cambridge University Press, 1990.

[69] X. Li and M. A. Scott. On the nesting theory of T-splines. Numerische Mathematik, submitted for publication, 2011.

[70] X. Li, J. Zheng, and T. W. Sederberg. On T-spline classification. Computer Aided Geometric Design, submitted for publication, 2011.

[71] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, and M. A. Scott. On the linear independence of T-splines. Computer Aided Geometric Design, submitted for publication, 2010.

[72] S. Lipton, J. A. Evans, Y. Bazilevs, T. Elguedj, and T. J. R. Hughes. Robustness of isogeometric structural discretizations under severe mesh distortion. Computer Methods in Applied Mechanics and Engineering, 199(5-8):357–373, 2010.

[73] D. Liu. GC^1 continuity conditions between two adjacent rational Bézier surface patches. Computer Aided Geometric Design, 7(1-4):151–163, 1990.

[74] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.

[75] C. Loop. Second order smoothness over extraordinary vertices. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 165–174. ACM, 2004.

[76] C. Loop and S. Schaefer. Approximating Catmull-Clark subdivision surfaces with bicubic patches. ACM Trans. Graph., 27:8:1–8:11, March 2008.

[77] C. Loop and S. Schaefer. G^2 tensor product splines over extraordinary vertices. In Computer Graphics Forum, volume 27, pages 1373–1382. Blackwell Science Ltd., 2008.

[78] G. G. Lorentz. Bernstein Polynomials. Chelsea Publishing Co., New York, 1986.

[79] J. Lu. Circular element: Isogeometric elements of smooth boundary. Computer Methods in Applied Mechanics and Engineering, 198(30-32):2391–2402, 2009.

[80] T. Martin, E. Cohen, and R. M. Kirby. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Computer Aided Geometric Design, 26(6):648–664, 2009.

[81] C. Miehe, M. Hofacker, and F. Welschinger. A phase-field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. Computer Methods in Applied Mechanics and Engineering, 199(45-48):2765–2778, 2010.

[82] C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. International Journal for Numerical Methods in Engineering, 83(10):1273–1311, 2010.

[83] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. International Journal for Numerical Methods in Engineering, 46(1):131–150, 1999.

[84] A. P. Nagy, M. M. Abdalla, and Z. Gurdal. Isogeometric sizing and shape optimization of beam structures. Computer Methods in Applied Mechanics and Engineering, 199(17-20):1216–1230, 2010.

[85] A. P. Nagy, M. M. Abdalla, and Z. Gurdal. On the variational formulation of stress constraints in isogeometric design. Computer Methods in Applied Mechanics and Engineering, 199(41-44):2687–2696, 2010.

[86] R. H. J. Peerlings, R. De Borst, W. A. M. Brekelmans, and J. H. P. De Vree. Gradient enhanced damage for quasi-brittle materials. International Journal for Numerical Methods in Engineering, 39(19):3391–3403, 1996.

[87] J. Peters. Patching Catmull-Clark meshes. In Proceedings of the 27th annual conference on computer graphics and interactive techniques, pages 255–258. ACM Press/Addison-Wesley Publishing Co., 2000.

[88] L. Piegl and W. Tiller. The NURBS Book. Springer-Verlag, New York, 1997.

[89] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. Computer Methods in Applied Mechanics and Engineering, 199(29-32):2059–2071, 2010.

[90] L. Ramshaw. Blossoms are polar forms. Computer Aided Geometric Design, 6(4):323–358, 1989.

[91] E. Rank, A. Düster, V. Nübel, K. Preusch, and O. T. Bruhns. High order finite elements for shells. Computer Methods in Applied Mechanics and Engineering, 194 (21-24):2494–2512, 2005.

[92] K. Ravi-Chandar. Dynamic fracture of nominally brittle materials. International Journal of Fracture, 90(1):83–102, 03 1998.

[93] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. Computer Aided Geometric Design, 12(2):153–174, 1995.

[94] U. Reif and P. Schröder. Curvature integrability of subdivision surfaces. Advances in Computational Mathematics, 14(2):157–174, 2001.

[95] J. J. C. Remmers, R. de Borst, and A. Needleman. The simulation of dynamic crack propagation using the cohesive segments method. Journal of the Mechanics and Physics of Solids, 56(1):70–92, 1 2008.

[96] D. F. Rogers. An Introduction to NURBS with Historical Perspective. Academic Press, San Diego, CA, 2001.

[97] Y. Saad. Iterative methods for sparse linear systems. PWS Pub. Co., Albany, NY, 1996.

[98] M. Sabin. Recent progress in subdivision: A survey. Advances in Multiresolution for Geometric Modelling, pages 203–230, 2005.

[99] M. A. Scott, M. J. Borden, C. V. Verhoosel, and T. J. R. Hughes. Isogeometric Finite Element Data Structures based on Bézier Extraction of T-splines.

International Journal for Numerical Methods in Engineering,, in press, doi: 10.1002/nme.3167, 2010.

[100] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes. Local refinement of analysis-suitable T-splines. Computer Methods in Applied Mechanics and Engineering, submitted for publication, 2010.

[101] M. A. Scott, T. W. Sederberg, P. Arner, L. Dedé, and T. J. R. Hughes. Analysis-suitable T-splines of arbitrary topology. in preparation, 2011.

[102] M. T. Sederberg and T. W. Sederberg. T-splines: A technology for marine design with minimal control points. `http://www.tsplines.com/technicalpapers.html`, 2010.

[103] T. W. Sederberg. Computer Aided Geometric Design: Course Notes. `http://www.tsplines.com/educationportal.html`, 2010.

[104] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. ACM Trans. Graph., 23:276–283, August 2004.

[105] T. W. Sederberg, G. T. Finnigan, X. Li, and H. Lin. Watertight trimmed NURBS. ACM Trans. Graph., 27:1–79, August 2008.

[106] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. ACM Trans. Graph., 22:477–484, July 2003.

[107] T. W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques, SIGGRAPH '98, pages 387–394, New York, NY, USA, 1998. ACM.

[108] T. W. Sederberg, J. Zheng, and X. Song. Knot intervals and multi-degree splines. Computer Aided Geometric Design, 20:455–468, 2003.

[109] X. Shi, T. Wang, P. Wu, and F. Liu. Reconstruction of convergent G^1 smooth B-spline surfaces. Computer Aided Geometric Design, 21(9):893–913, 2004.

[110] L. J. Sluys and R. de Borst. Dispersive properties of gradient-dependent and rate-dependent media. Mechanics of Materials, 18(2):131–149, 1994.

[111] S.-H. Song, H. Wang, and T. Belytschko. A comparative study on finite element methods for dynamic fracture. Computational Mechanics, 42(2):239–250, July 2008.

[112] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 395–404. ACM, 1998.

[113] T-Splines, Inc. http://www.tsplines.com/products/t-tools-libraries.html, 2011.

[114] T-Splines, Inc. http://www.tsplines.com/rhino/, 2011.

[115] C. V. Verhoosel, M. A. Scott, M. J. Borden, R. de Borst, and T. J. R. Hughes. Isogeometric failure analysis. Recent Developments and Innovative Applications in Computational Mechanics, pages 275–281, 2011.

[116] C. V. Verhoosel, M. A. Scott, R. de Borst, and T. J. R. Hughes. An isogeometric approach to cohesive zone modeling. International Journal for Numerical Methods in Engineering,, in press, doi: 10.1002/nme.3061, 2010.

[117] C. V. Verhoosel, M. A. Scott, T. J. R. Hughes, and de Borst, R. An isogeometric analysis approach to gradient damage models. International Journal for Numerical Methods in Engineering,, in press, doi: 10.1002/nme.3150, 2010.

[118] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. Computer Methods in Applied Mechanics and Engineering, 197:2976–2988, 2008.

[119] W. Wang and Y. Zhang. Wavelets-based NURBS simplification and fairing. Computer Methods in Applied Mechanics and Engineering, 199(5-8):290–300, 2010.

[120] W. Wang, Y. Zhang, M. A. Scott, and T. J. R. Hughes. Converting an unstructured quadrilateral mesh to a standard T-spline surface. Computational Mechanics,, accepted for publication, 2011.

[121] Y. Zhang, Y. Bazilevs, S. Goswami, C. Bajaj, and T. J. R. Hughes. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Computer Methods in Applied Mechanics and Engineering, 196:2943–2959, 2007.

[122] J. Zheng, G. Wang, and Y. Liang. Curvature continuity between adjacent rational Bézier patches. Computer Aided Geometric Design, 9(5):321–335, 1992.